

# **DAC Linearity Improvement Algorithm With Unit Cell Sorting Based on Magic Square**

**Masashi Higashino**

**Shaiful Nizam Mohyar,** Haruo Kobayashi

**Division of Electronics and Informatics**

**Gunma University, Japan**

**Universiti Malaysia Perlis, Malaysia**

# OUTLINE

- Research Objective
- Current Steering DAC
- What is Magic Square ?
- Proposed Algorithm
- Simulation Results
- Conclusion

# OUTLINE

- Research Objective
- Current Steering DAC
- What is Magic Square ?
- Proposed Algorithm
- Simulation Results
- Conclusion

# Research Objective

## Research Background

- Demand for DAC in communication systems
  - High linearity
  - High spurious free dynamic range (SFDR)

## Our Approach

- Unary DAC linearity improvement
  - Unit cell sorting algorithm
  - Based on **Magic Square**
  - Digital method

No analog part modification

New!!!

# OUTLINE

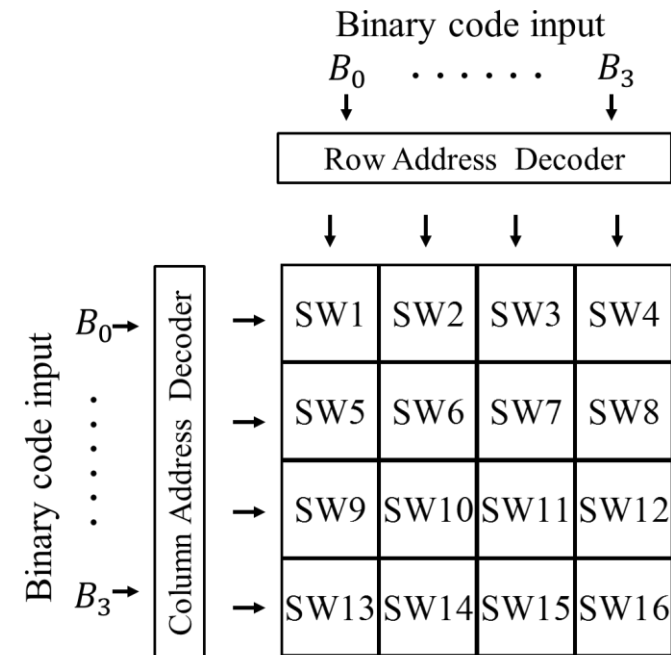
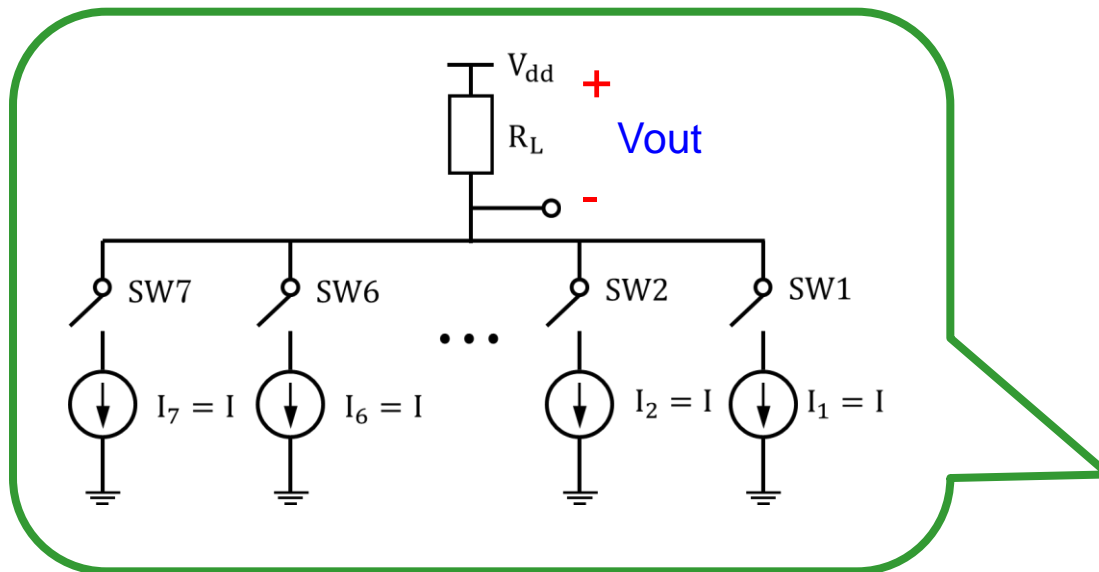
- Research Objective
- **Current Steering DAC**
- What is Magic Square ?
- Proposed Algorithm
- Simulation Results
- Conclusion

# Circuit and Features of Unary Current-Steering DAC

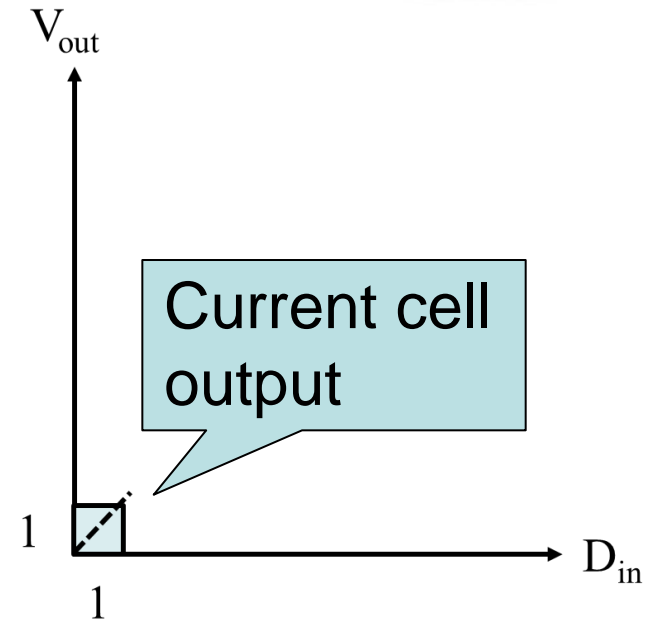
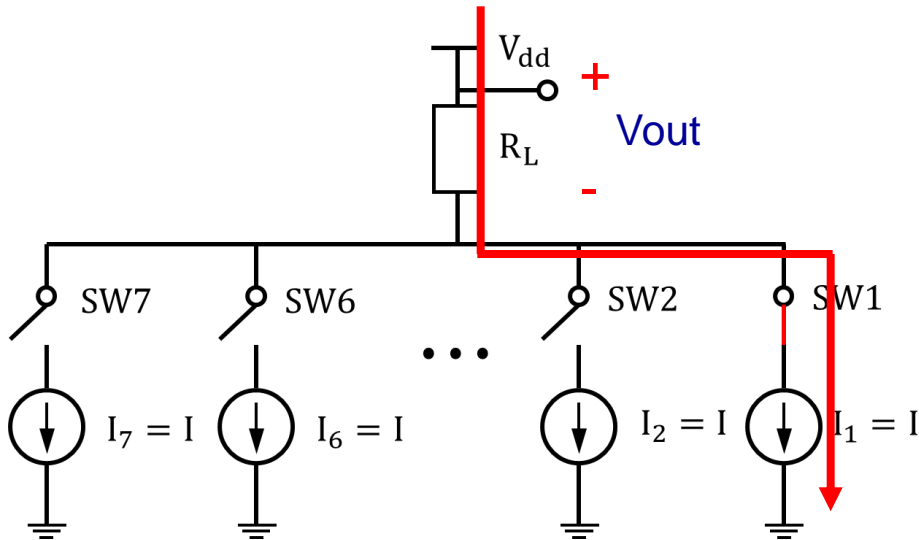
- Identical current sources
- Small glitch
- Inherent monotonicity
- High speed



- Large circuits
  - Decoder
  - Many switches and current sources



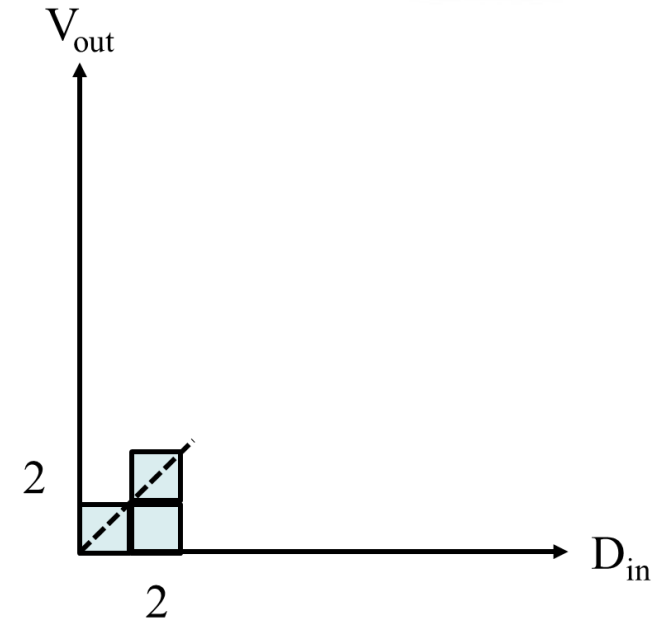
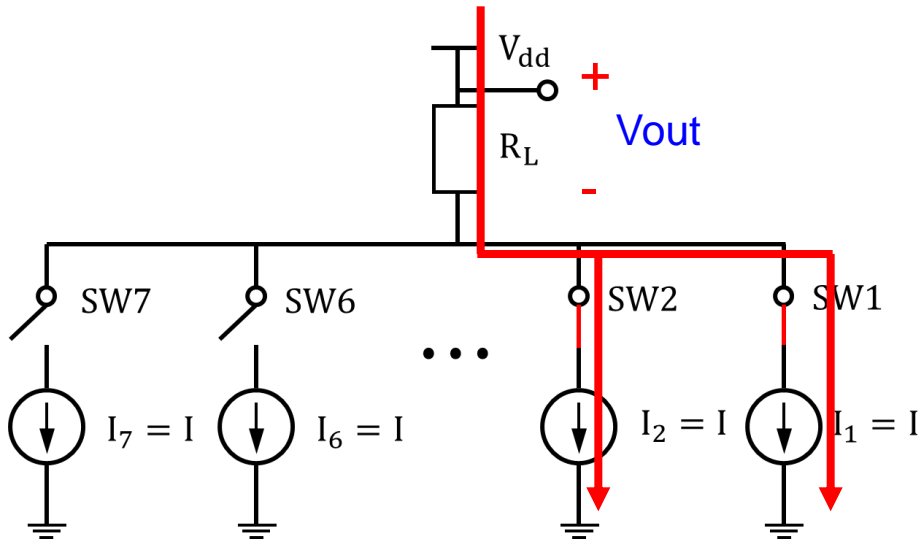
# Operation of Current Steering DAC (1)



Digital input = 1 , 1 current source.

$$V_{out} = R_L I_1$$

# Operation of Current Steering DAC (2)

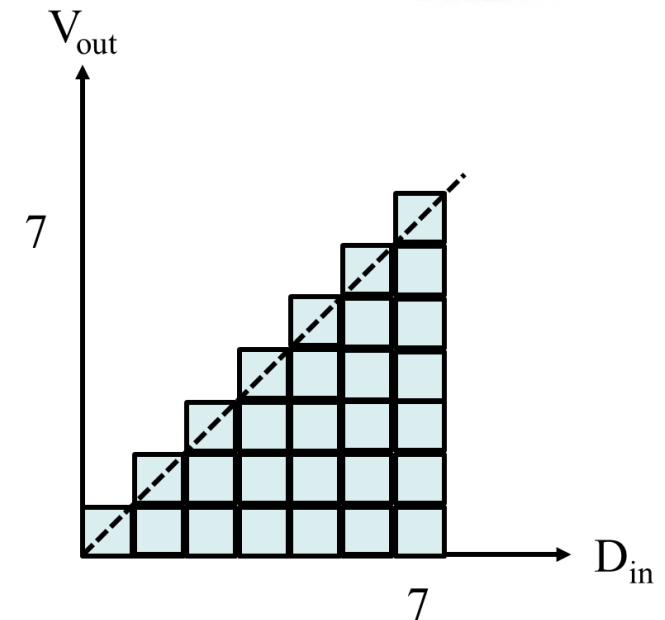
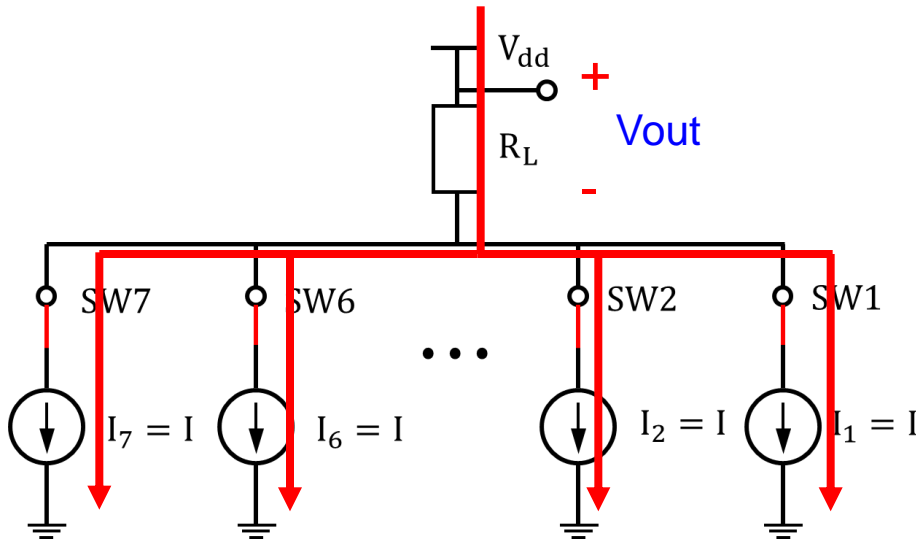


Digital input = 1 , 1 current source.  
 “ = 2 , 2 current sources.

$$V_{out} = R_L(I_1 + I_2)$$



# Operation of Current Steering DAC (3)



Digital input = 1 , 1 current source.  
 “ = 2 , 2 current sources.  
 “ = 7 , 7 current sources.

$$V_{out} = R_L(I_1 + I_2 + \dots + I_7)$$

$$I_1 = I_2 = \dots = I_6 = I_7$$

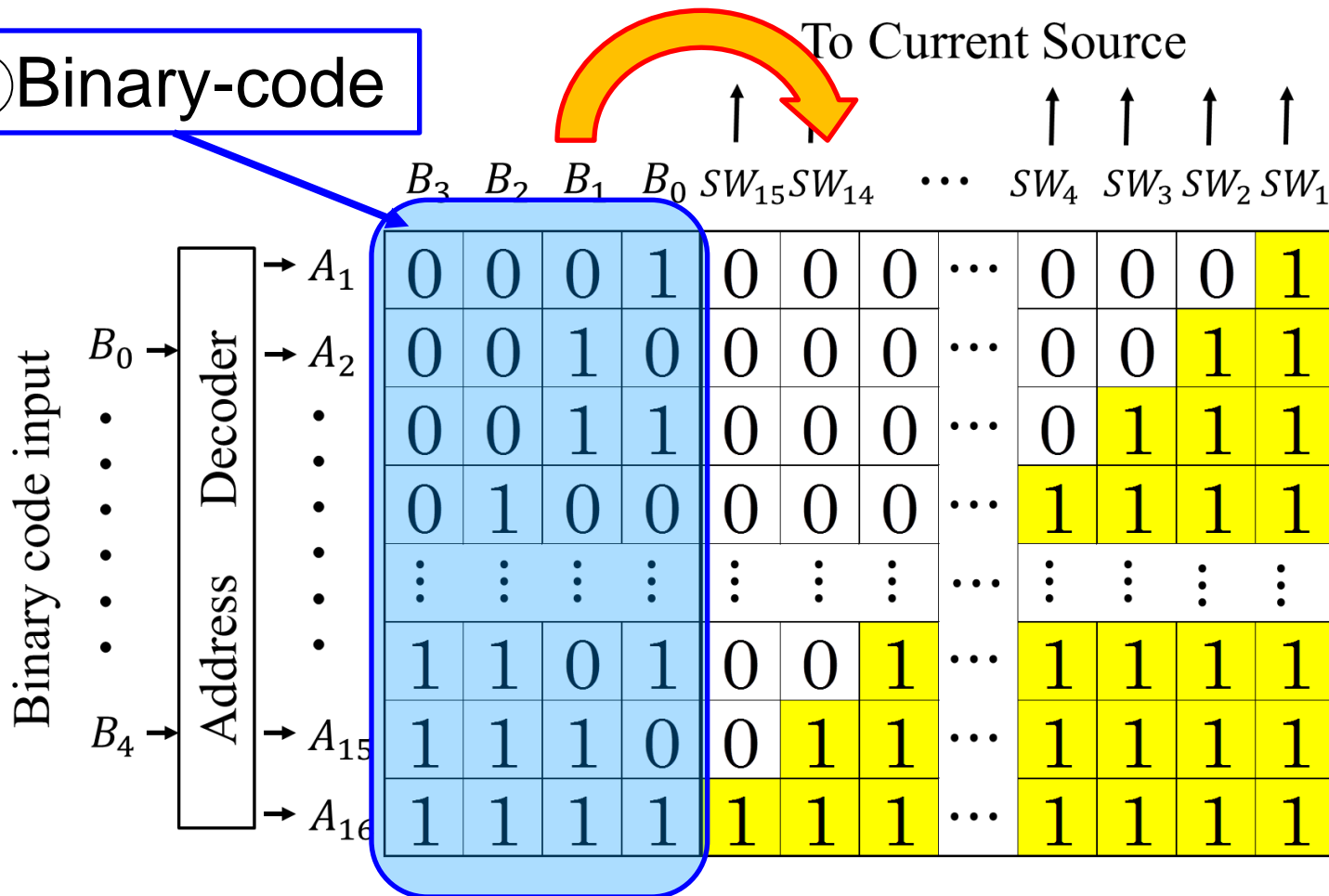


DAC is perfectly linear

# Conventional Unary DAC Decoder

## ② Thermometer-code

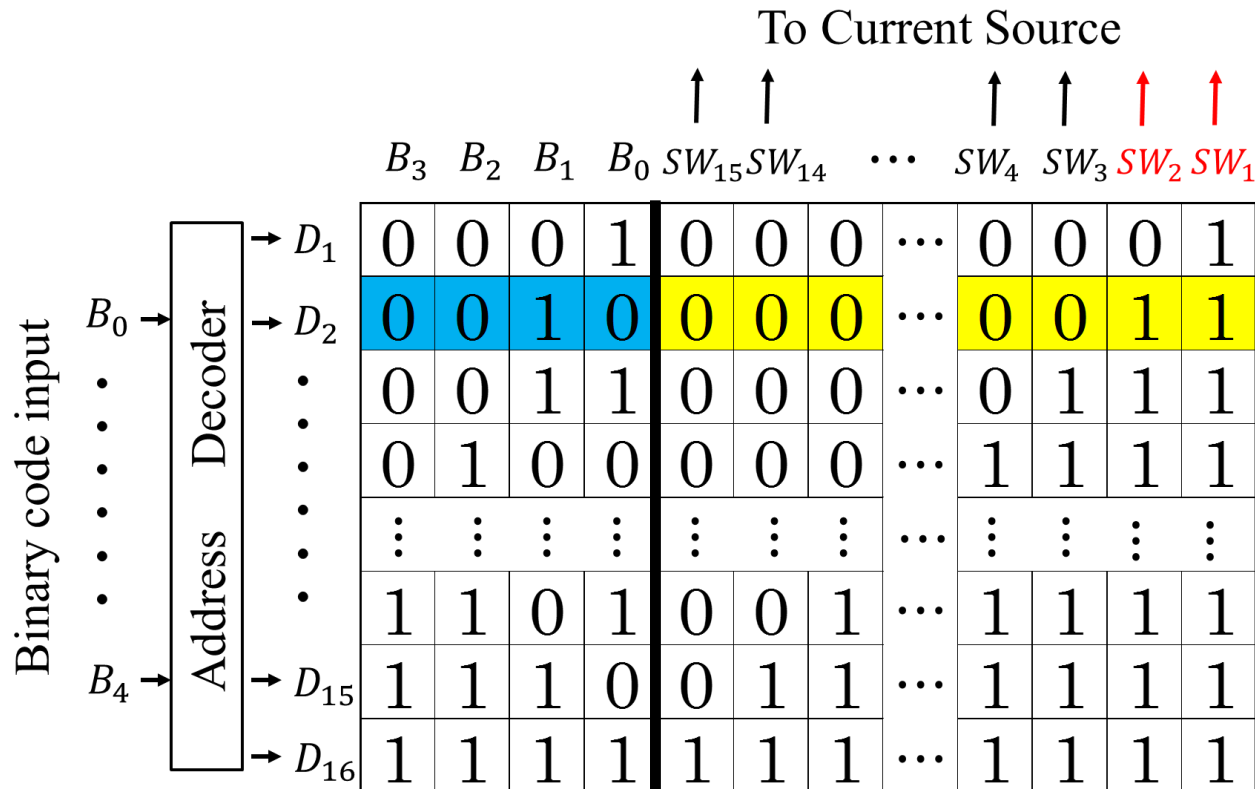
## ① Binary-code



# Operation of Unary DAC Decoder

## Example 1

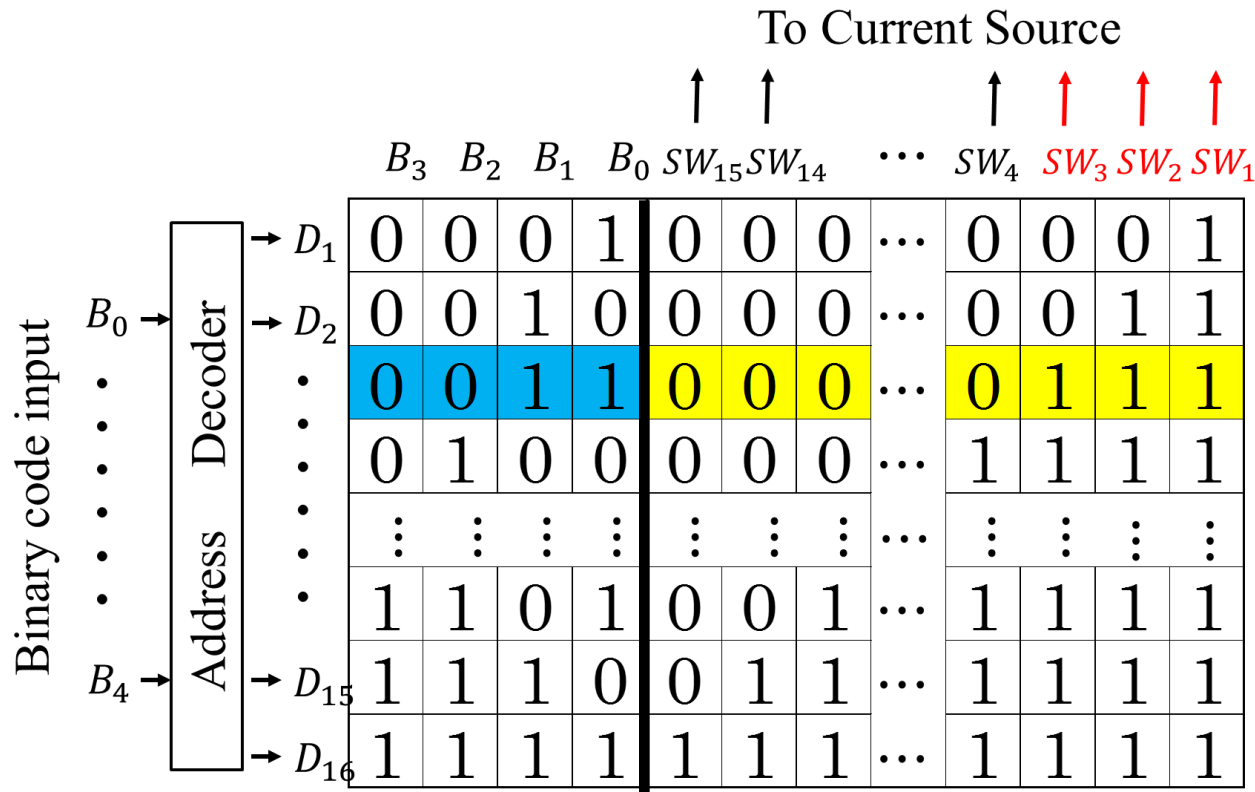
- Digital binary input (00**10**)
  - ➡ Thermometer code (0000 0000 0000 00**11**)
  - ➡ **2** current cells turn on.



# Operation of Unary DAC Decoder

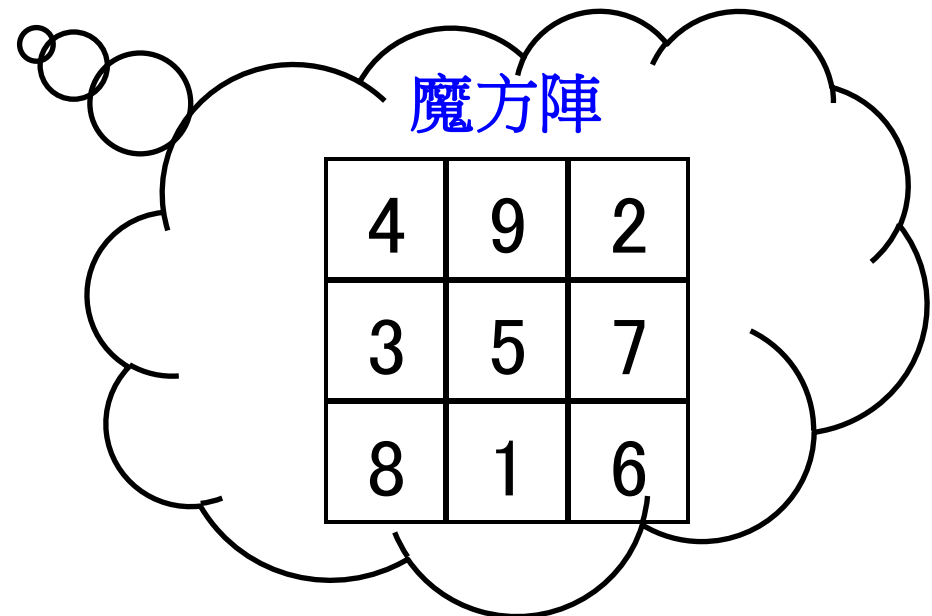
## Example 2

- Digital binary input (00**11**)
  - ➡ Thermometer code (0000 0000 0000 0**111**)
  - ➡ 3 current cells turn on.



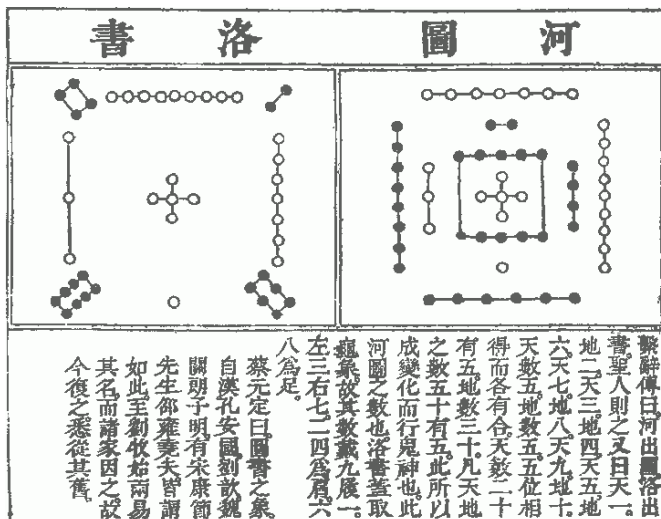
# OUTLINE

- Research Objective
- Current Steering DAC
- What is Magic Square ?
- Proposed Algorithm
- Simulation Results
- Conclusion



# What is Magic Square ?

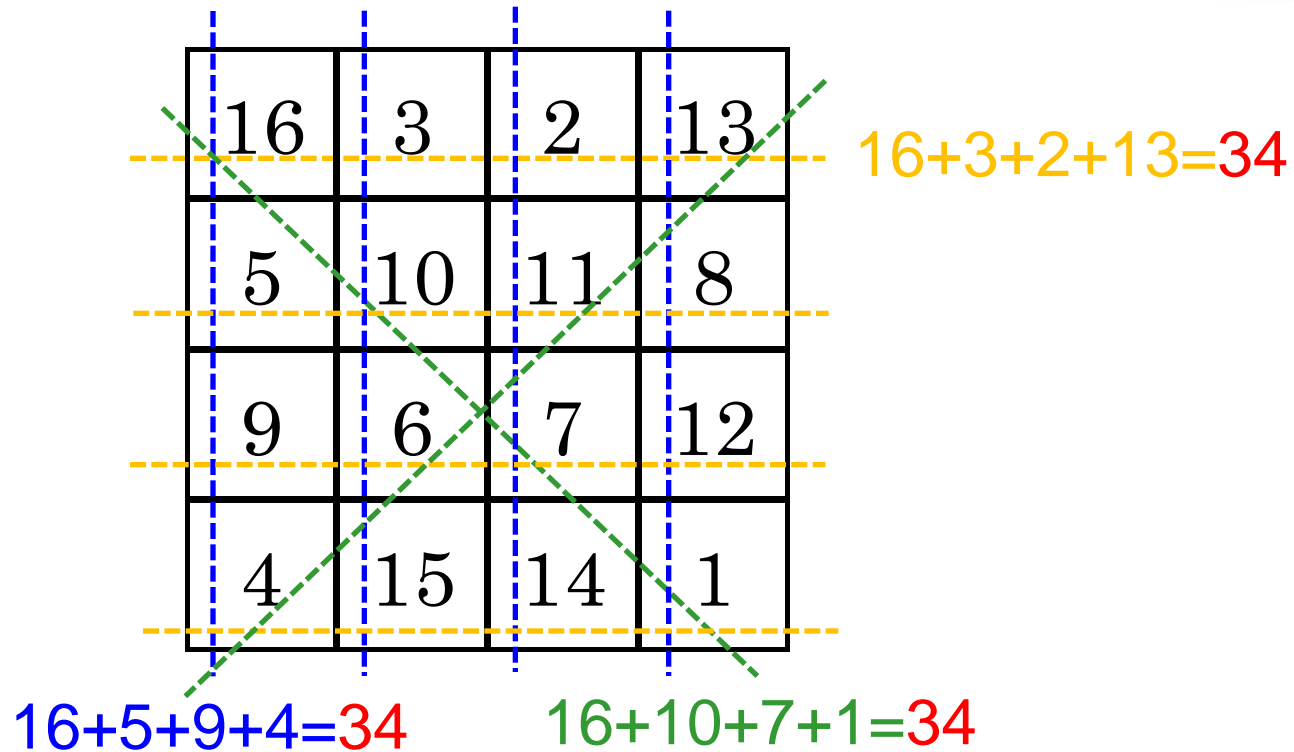
- Classical mathematics
- Origin from Chinese academia
- “Constant sum” characteristics
- Varieties of magic squares



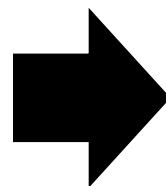
3x3 魔方陣

4	9	2
3	5	7
8	1	6

# Features of Magic Square



- Constant Sum
  - Row, column, diagonal



魔方陣 is Good balance

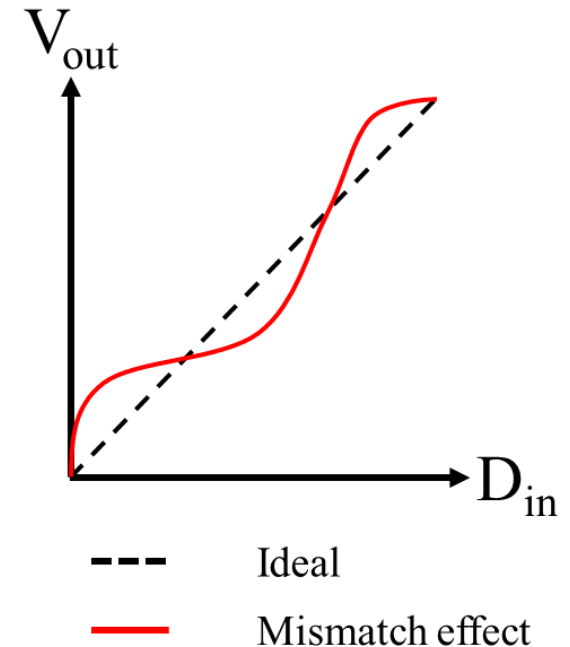
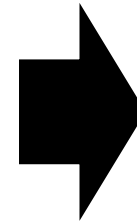
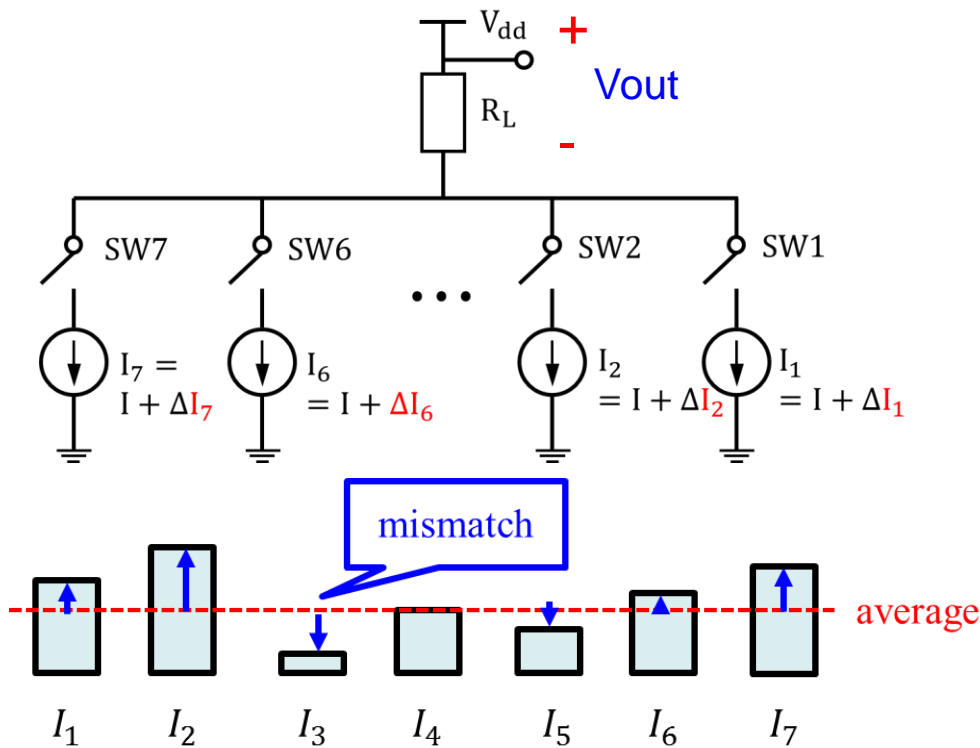


# OUTLINE

- Research Objective
- Current Steering DAC
- What is Magic Square ?
- **Proposed Algorithm**
- Simulation Results
- Conclusion



# Unit Current Source Mismatch Problem

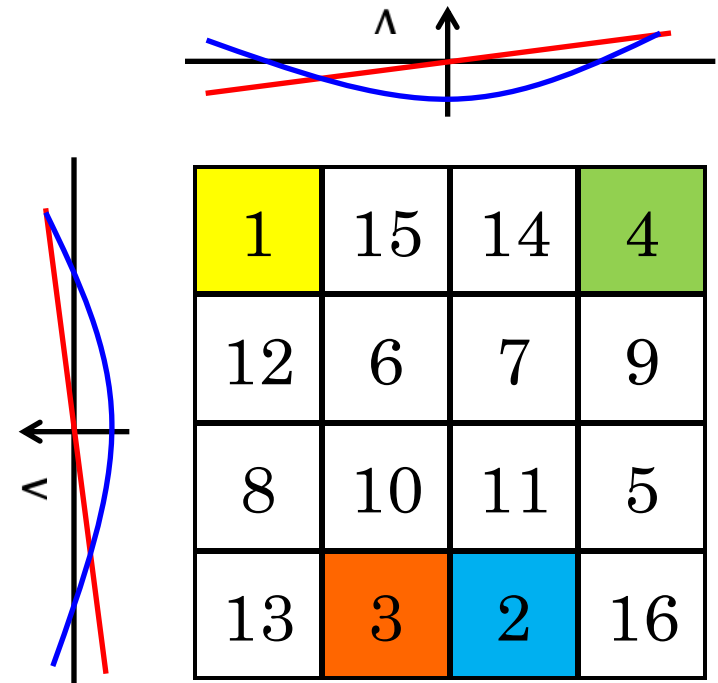
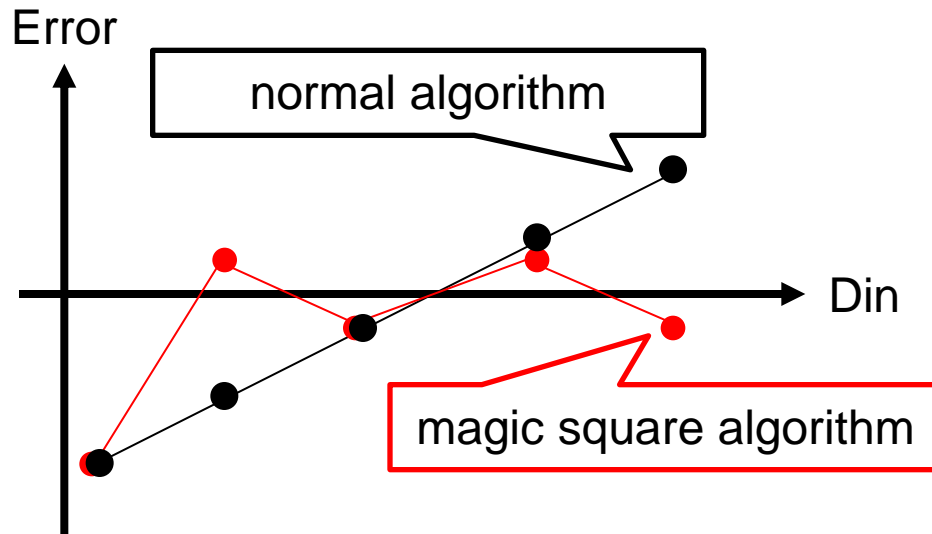


In practice, current sources have mismatches.

➡ **DAC becomes non-linear.**

# Possibility of Using Magic Square

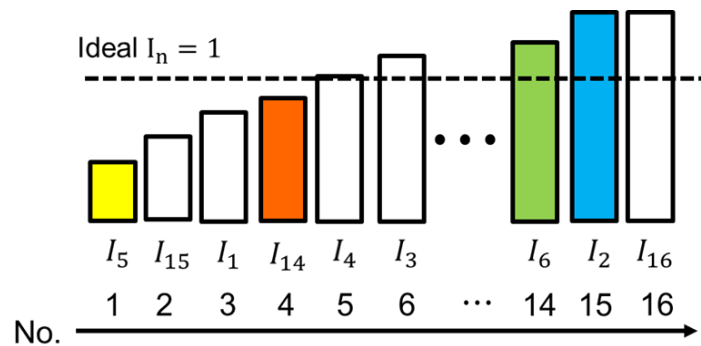
- Semiconductor devices have random and systematic mismatches
- Changing the switching order  
➡ Cancellation of mismatch effects
- We propose magic square algorithm



# Inspired New Algorithm

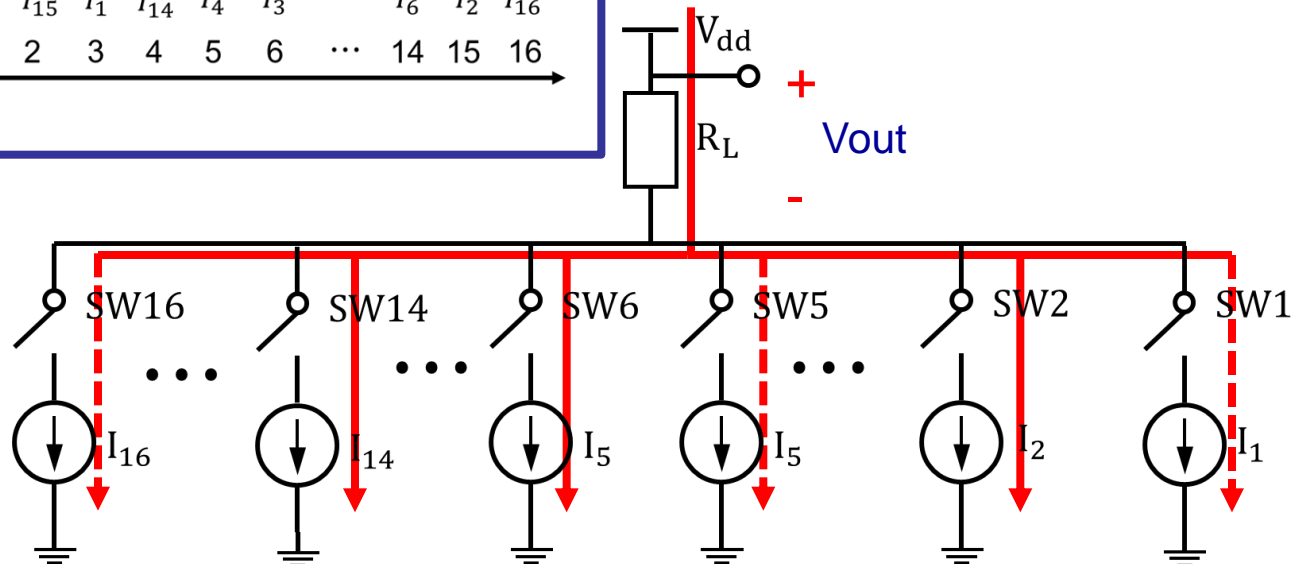
- Unit current source selection-order change algorithm
  - Mismatch effect cancellation

$I_1$	$I_2$	$I_3$	$I_4$
$I_5$	$I_6$	$I_7$	$I_8$
$I_9$	$I_{10}$	$I_{11}$	$I_{12}$
$I_{13}$	$I_{14}$	$I_{15}$	$I_{16}$



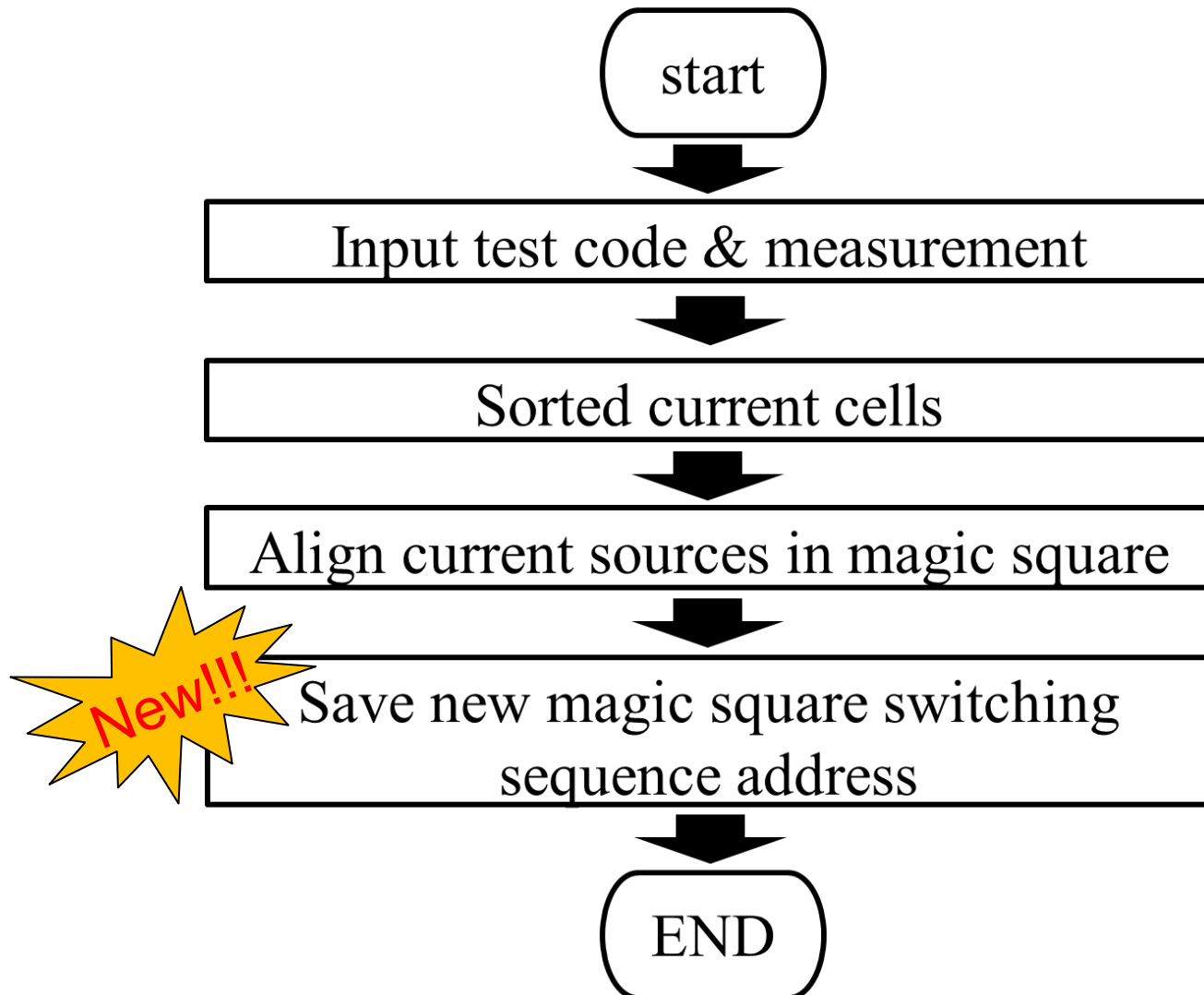
1. Measure the order of unit current cells
2. Align them virtually in magic square
3. Select current cells

1	15	14	4
12	6	7	9
8	10	11	5
13	3	2	16



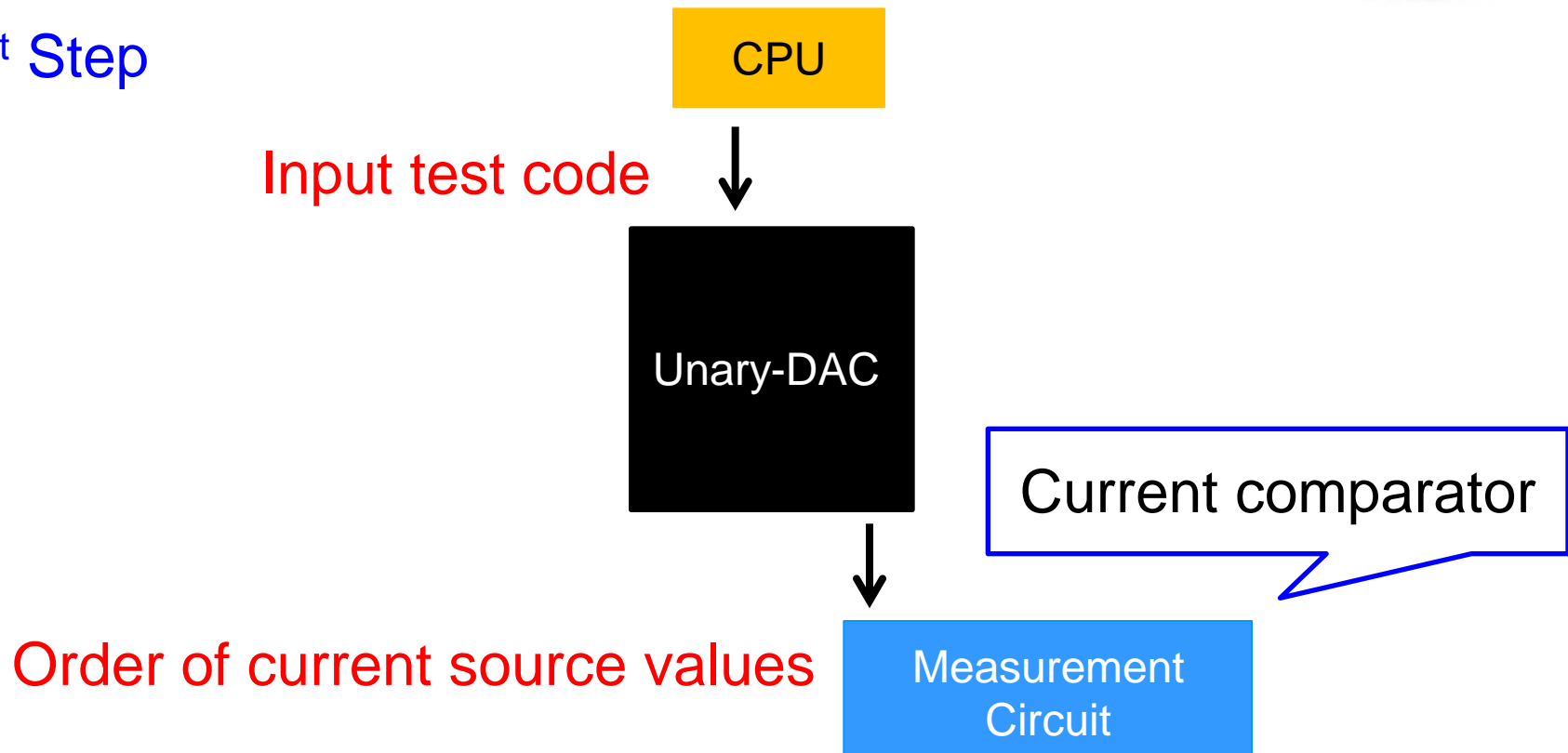
# Proposed DAC Non-linearity

## Calibration Algorithm



## Input Test Code & Measurement

1<sup>st</sup> Step



- CPU => input test code to unary-DAC cells
- Measurement circuit => order of current source values

## Measure Order of Current Cells

### 1<sup>st</sup> Step

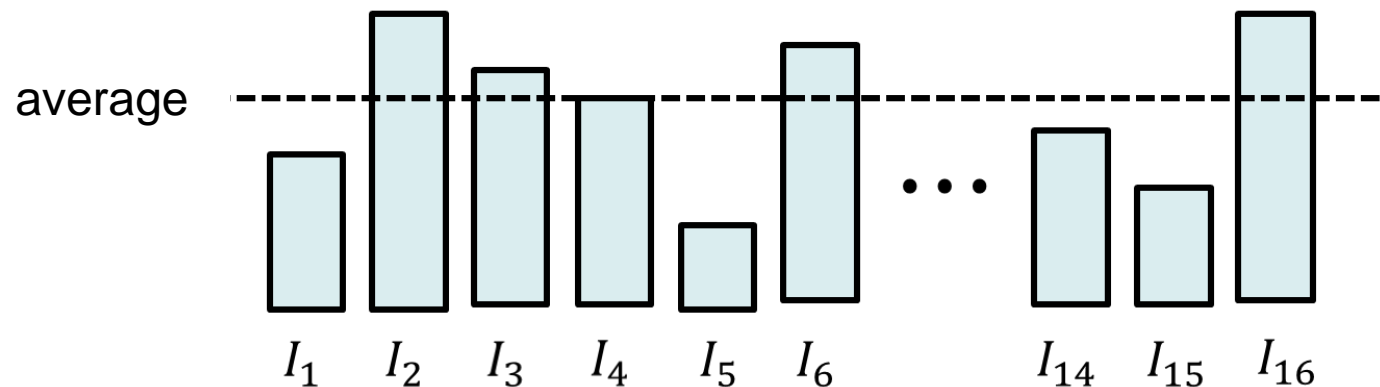
- Measure the order of current cell values by a current comparator.
- Not need accurate value measurement.

### 4-bit case

Current cell

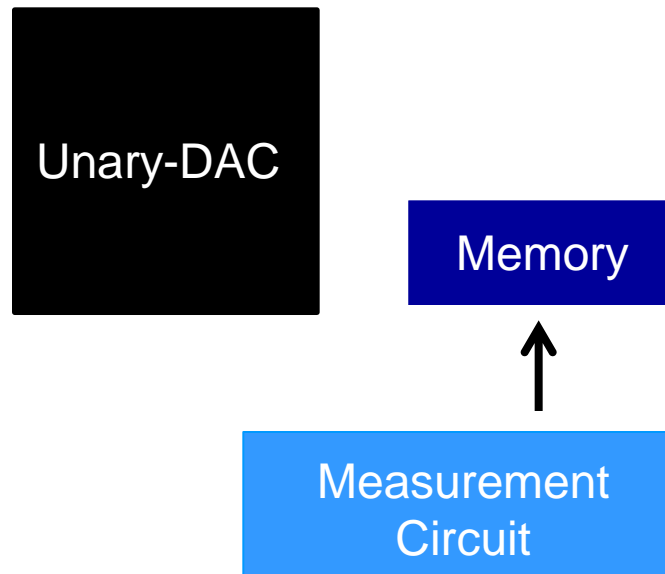
$I_1$	$I_2$	$I_3$	$I_4$
$I_5$	$I_6$	$I_7$	$I_8$
$I_9$	$I_{10}$	$I_{11}$	$I_{12}$
$I_{13}$	$I_{14}$	$I_{15}$	$I_{16}$

Original Current Source



## Unit Current Source Sorting

2<sup>nd</sup> step

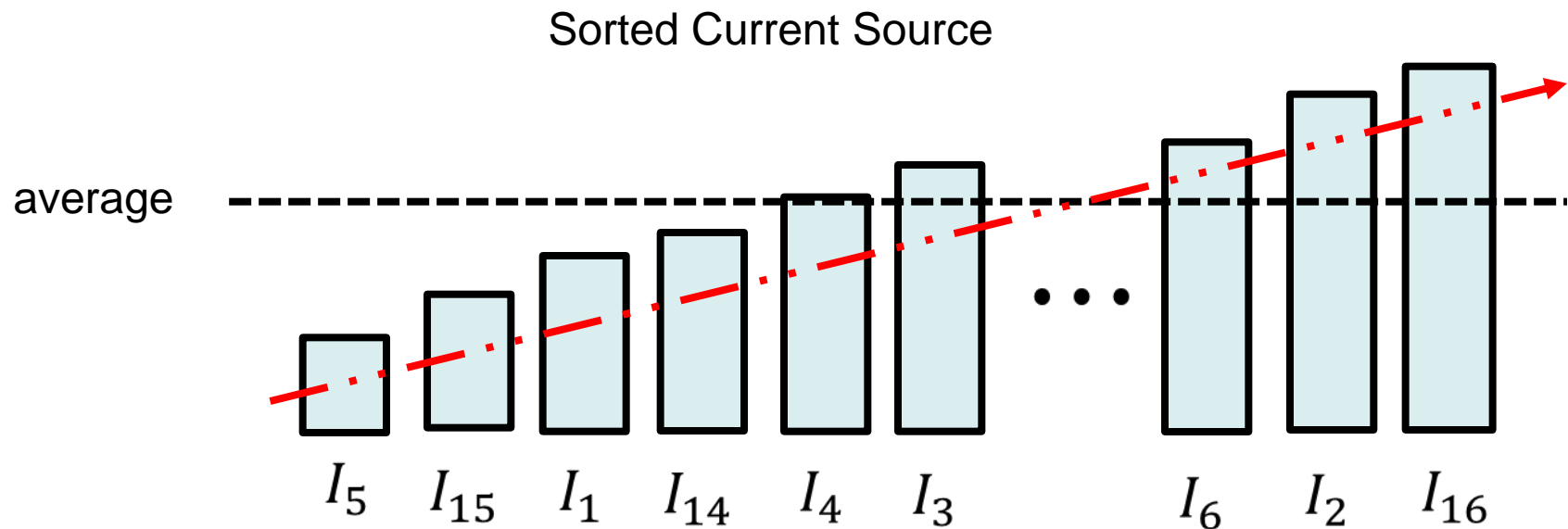


Sort and store the measured order of the unit current cell values into memory.

## Unit Current Source Sorting

2<sup>nd</sup> step

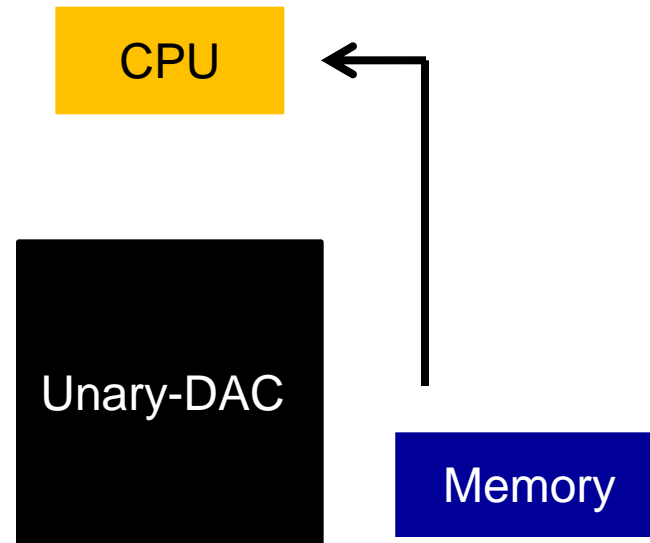
- Sort current source cells ascendingly.
- Store their information of cells number and value into memory.





## Current Source Sorting Based on Magic Square

3<sup>rd</sup> step



- Re-sort of current source values based on magic square

## Current Source Sorting Based on Magic Square (1)

### 3<sup>rd</sup> step

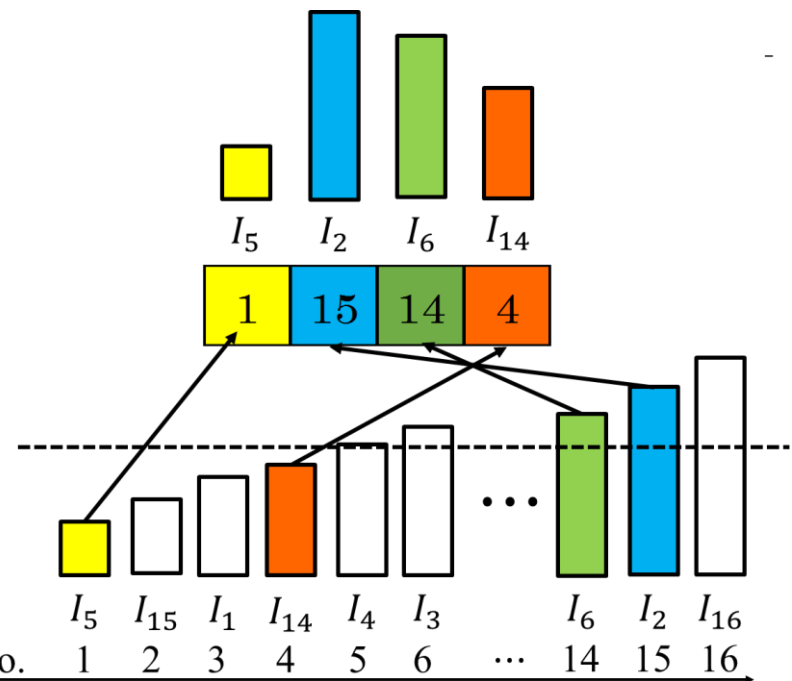
- Re-sorted of current source values based on magic square
- Store its info in decoder look-up table

$I_1$	$I_2$	$I_3$	$I_4$
$I_5$	$I_6$	$I_7$	$I_8$
$I_9$	$I_{10}$	$I_{11}$	$I_{12}$
$I_{13}$	$I_{14}$	$I_{15}$	$I_{16}$

switching order

1	15	14	4
12	6	7	9
8	10	11	5
13	3	2	16

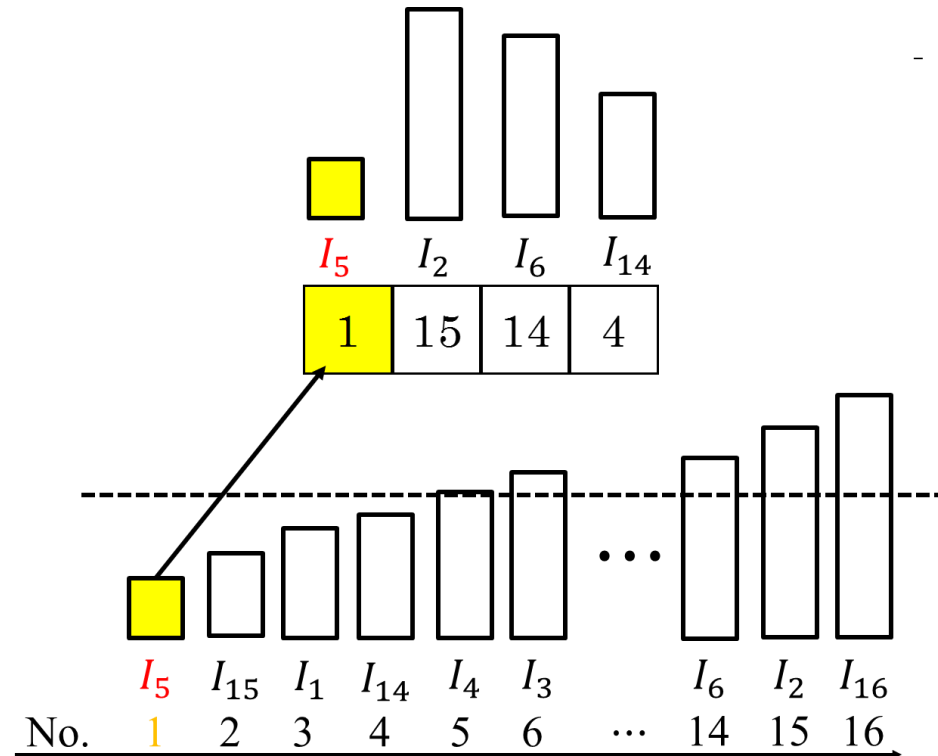
No.



## Current Source Sorting Based on Magic Square (2)

- Digital binary input (000**1**)  
➔ 1 current cells turn on

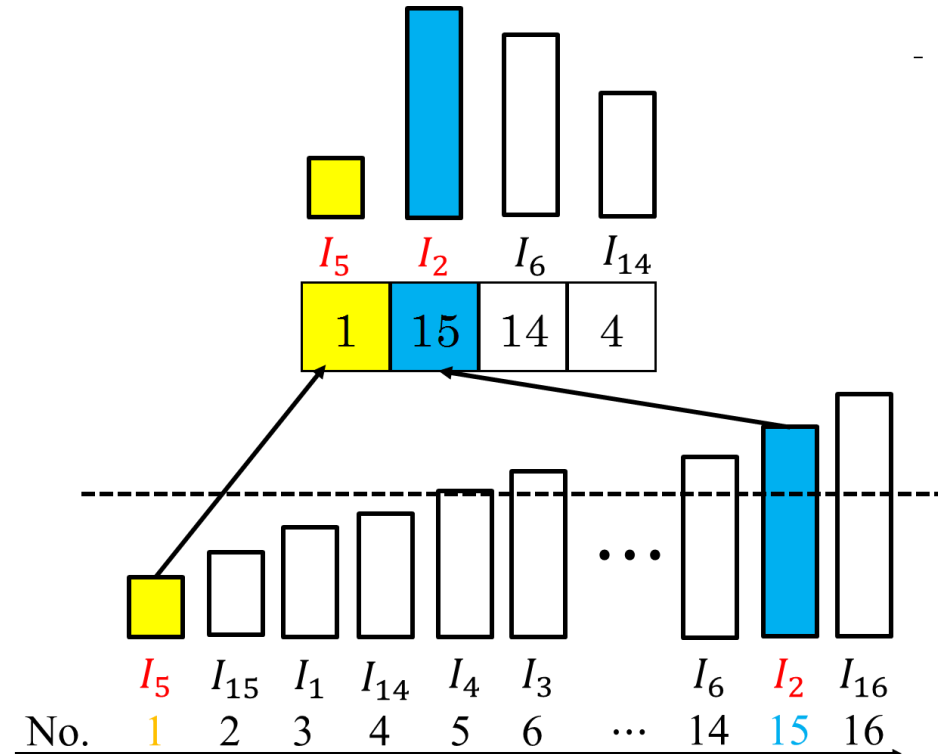
$I_1$	$I_2$	$I_3$	$I_4$
$I_5$	$I_6$	$I_7$	$I_8$
$I_9$	$I_{10}$	$I_{11}$	$I_{12}$
$I_{13}$	$I_{14}$	$I_{15}$	$I_{16}$



## Current Source Sorting Based on Magic Square (3)

- Digital binary input (00**10**)  
➔ 2 current cells turn on

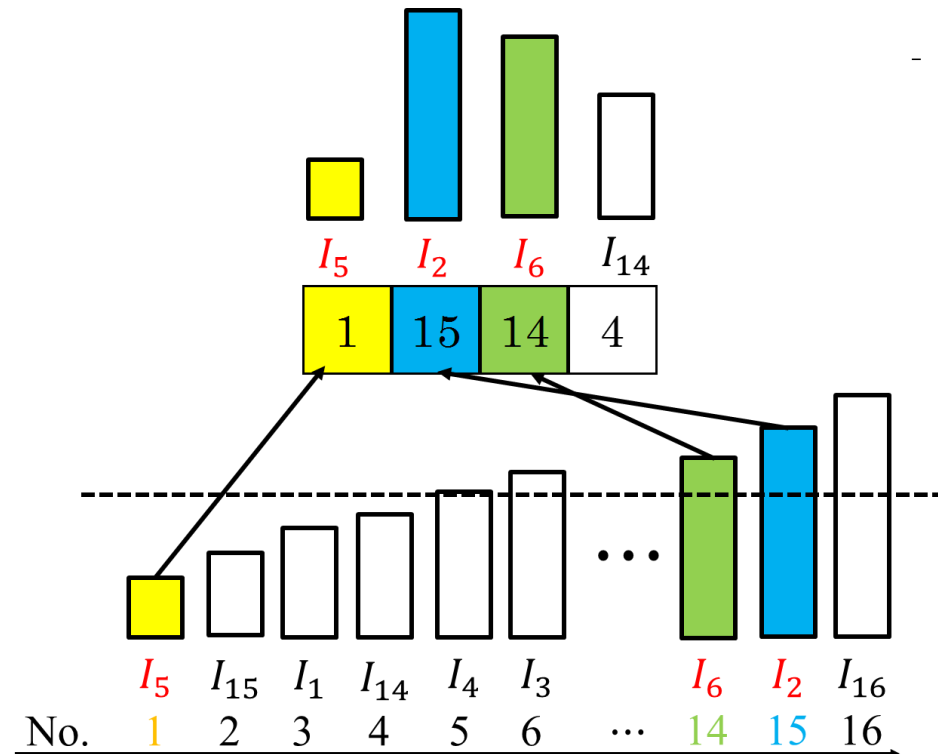
$I_1$	$I_2$	$I_3$	$I_4$
$I_5$	$I_6$	$I_7$	$I_8$
$I_9$	$I_{10}$	$I_{11}$	$I_{12}$
$I_{13}$	$I_{14}$	$I_{15}$	$I_{16}$



## Current Source Sorting Based on Magic Square (4)

- Digital binary input (00**11**)  
➔ 3 current cells turn on

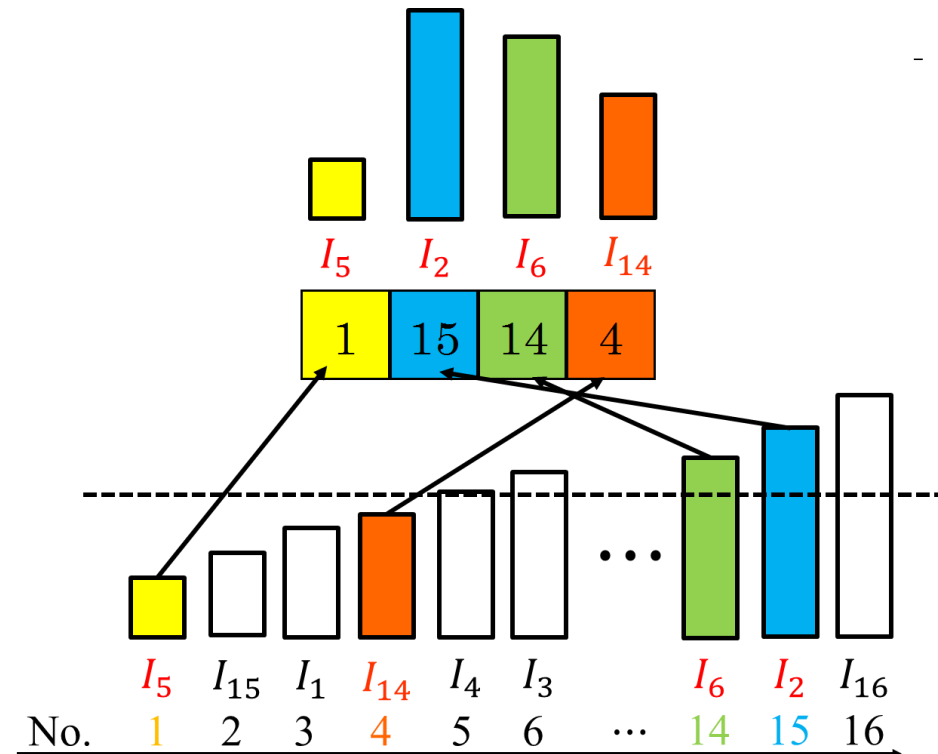
$I_1$	$I_2$	$I_3$	$I_4$
$I_5$	$I_6$	$I_7$	$I_8$
$I_9$	$I_{10}$	$I_{11}$	$I_{12}$
$I_{13}$	$I_{14}$	$I_{15}$	$I_{16}$



## Current Source Sorting Based on Magic Square (5)

- Digital binary input (0**1**00)  
➔ 4 current cells turn on

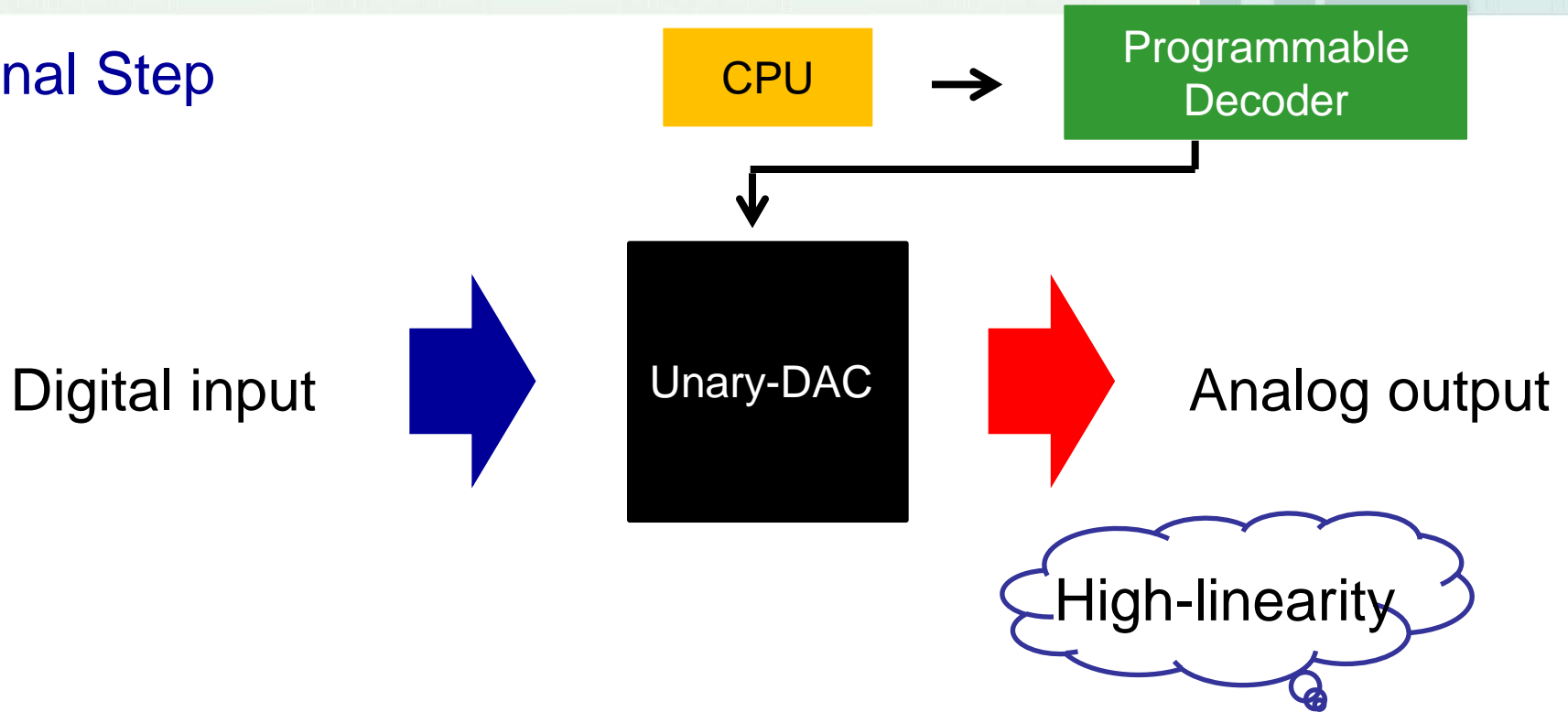
$I_1$	$I_2$	$I_3$	$I_4$
$I_5$	$I_6$	$I_7$	$I_8$
$I_9$	$I_{10}$	$I_{11}$	$I_{12}$
$I_{13}$	$I_{14}$	$I_{15}$	$I_{16}$



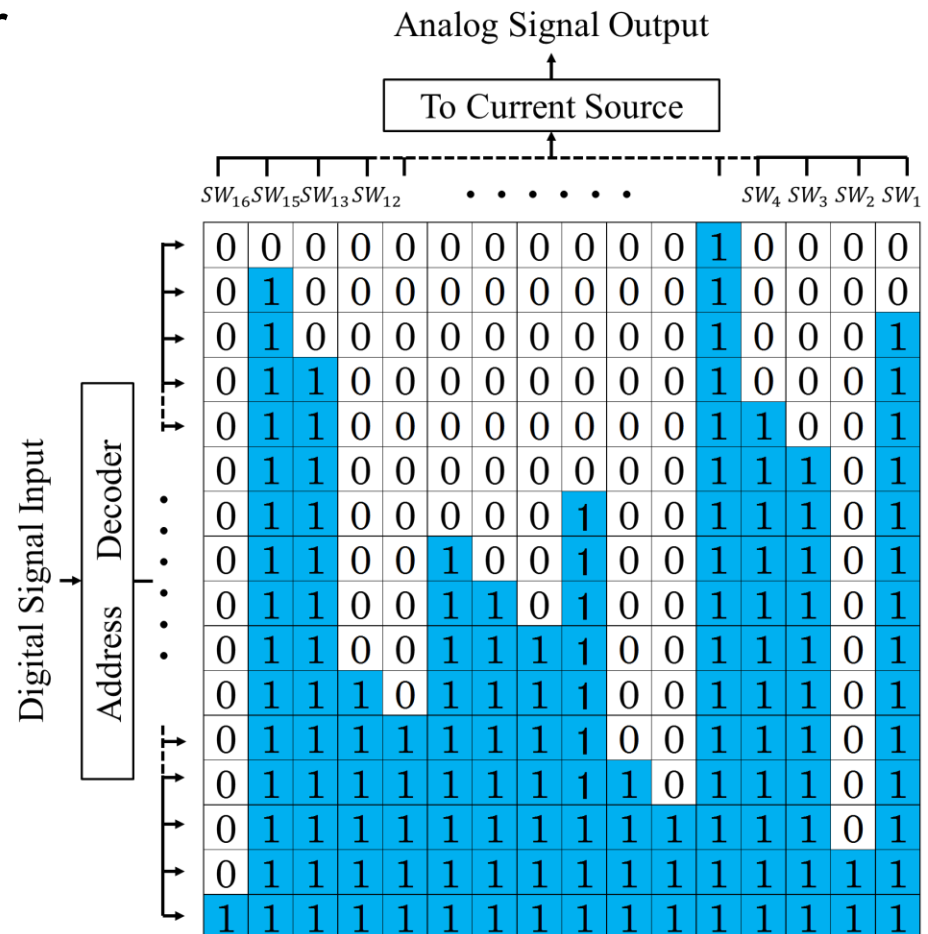
# Proposed Algorithm

## LUT-Magic Square Decoder

Final Step



Store switching sequence based on magic square into programmable decoder.





# OUTLINE

- Research Objective
- Current Steering DAC
- What is Magic Square ?
- Proposed Algorithm
- **Simulation Results**
- Conclusion

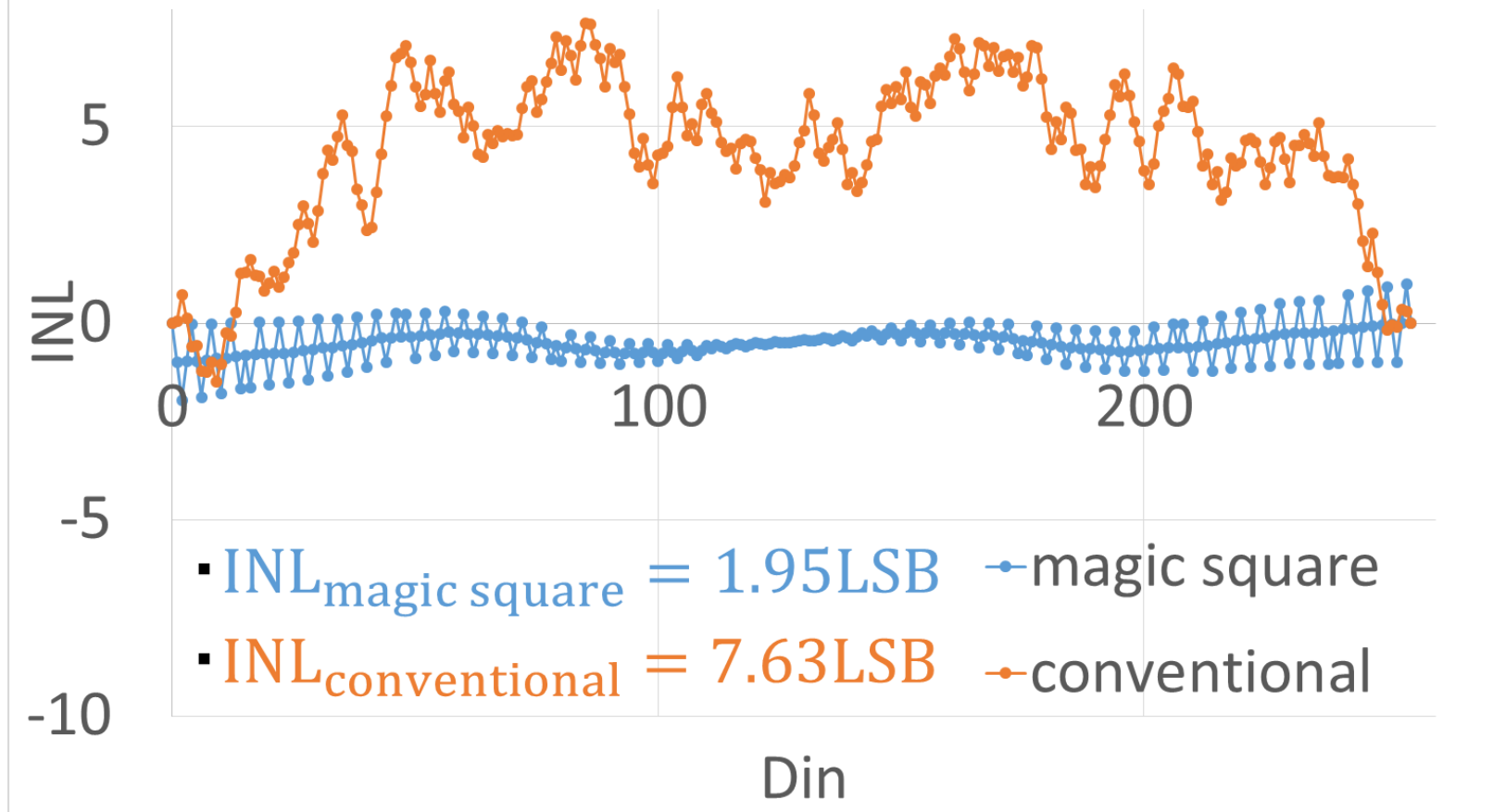
# Simulation Conditions

- MATLAB simulation
- 8-bit unary DAC
  - Static performance (INL, DNL)
  - Dynamic performance (SFDR)
- Compared two methods
  - Conventional thermometer-code decoder usage
  - Proposed magic-square-based algorithm
- Mismatch of current sources
  - Current sources have average of value 1.0
  - Random number between  $-1 < \text{mismatch} < +1$  (uniform distribution)

## - Static Performance INL -

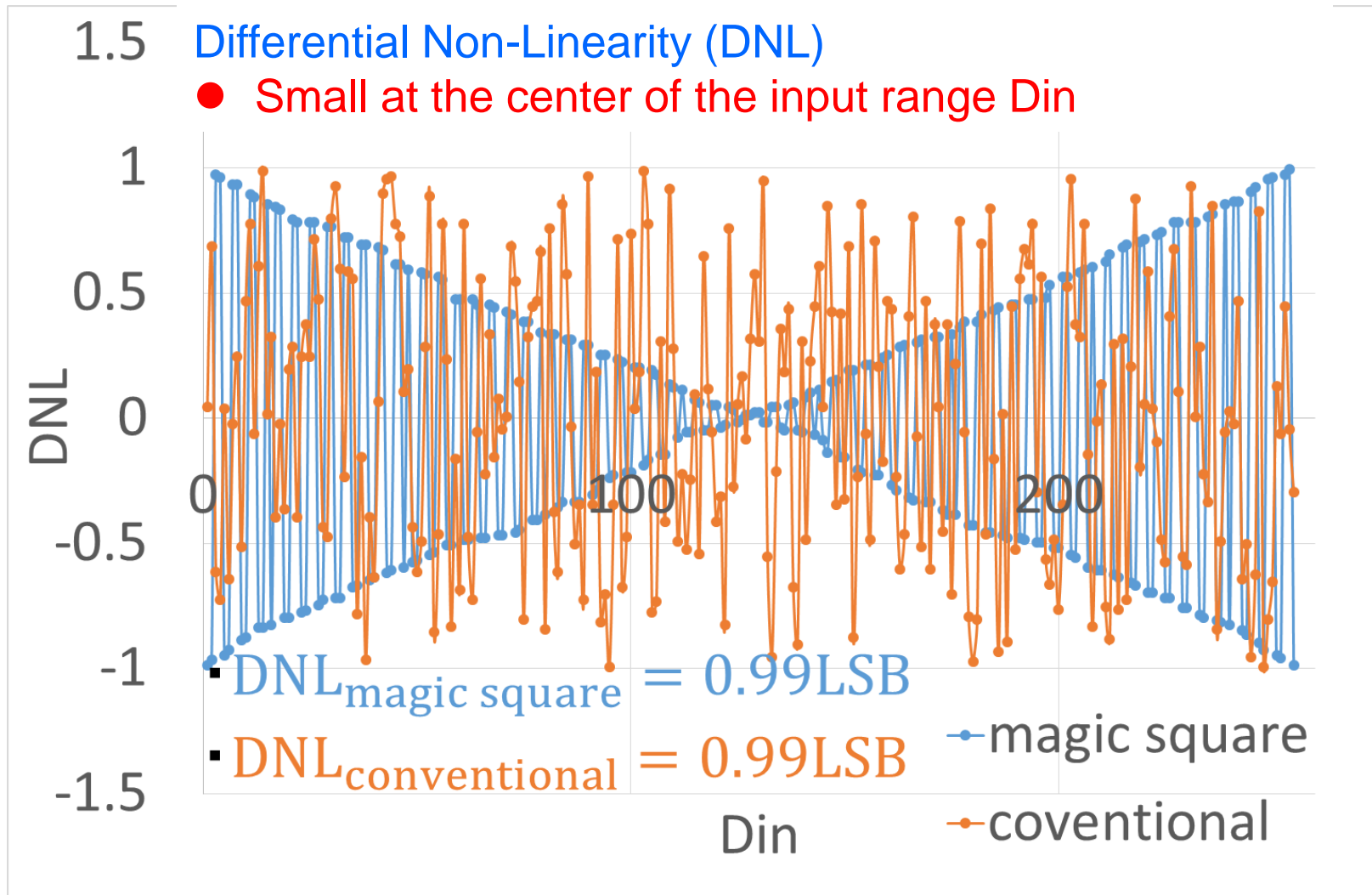
### Integral Non-Linearity (INL)

- 5.7 LSB improvement by the magic square algorithm
- 0.0 LSB at the center of the input range  $D_{in}$



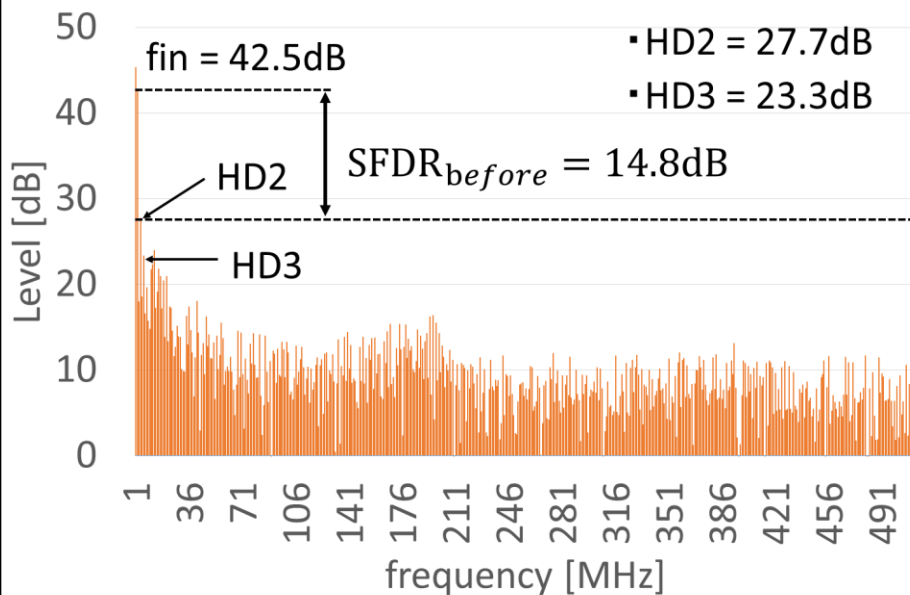
# Simulation Result

## - Static Performance DNL -

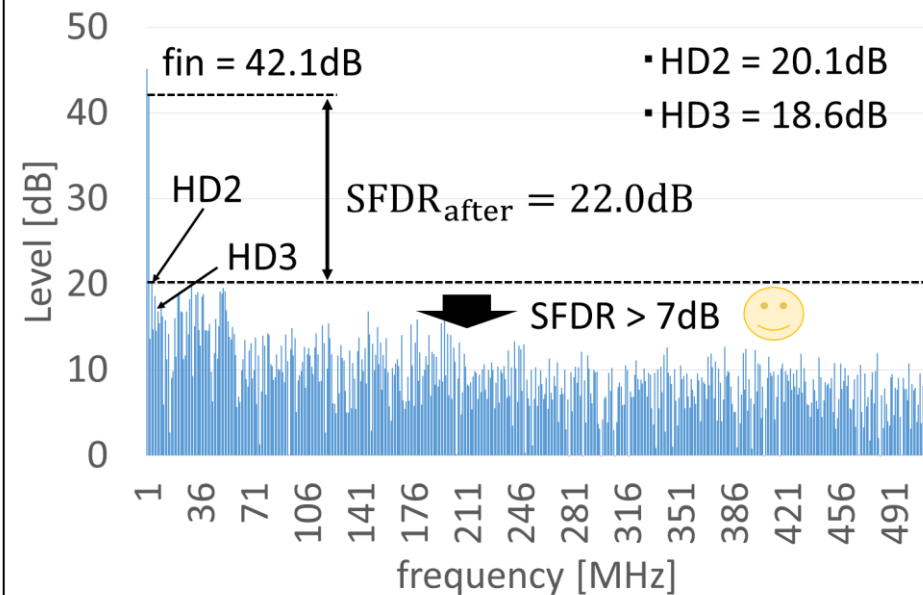


## - Dynamic Performance SFDR -

### ● Conventional method



### ● Proposed method



● SFDR improvement by 7 dB

# OUTLINE

- Research Objective
- Current Steering DAC
- What is Magic Square ?
- Proposed Algorithm
- Simulation Results
- Conclusion

## Conclusion

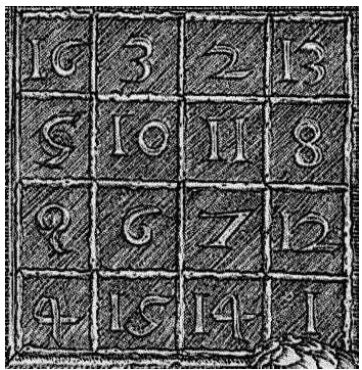
- Unary DAC linearity improvement
  - Cancel unit current cell mismatch effects
  - Unit current cell selection algorithm
    - ➡ Digital method
  - Based on magic square
  - Measurement of the order of current cell values
- MATLAB simulation
  - INL , DNL improvement
    - at the center of the input range.
  - SFDR improvement



# Final Statement

## 温故知新

Classical mathematics can contribute modern technology.







IEEE International Symposium on Intelligent Signal  
Processing and Communication Systems 2017

NOVEMBER 6-9, 2017, XIAMEN, CHINA



Nov. 8 WP-L6 16:10-17:40

# DAC Linearity Improvement With Layout Technique Using Magic and Latin Squares

Dan Yao, Yifei Sun, M. Higashino, S. N. Mohyar T. Yanagida  
T. Arafune, N. Tsukiji, H. Kobayashi

*Gunma University*



Kobayashi Lab.  
Gunma University

# Contents

---

- Research Objective
- Segment Type DA Converter
- Characteristic of Variation in Circuit Element
- Proposed Layout Method
  - Magic Square
  - Latin Square
- Conclusion

# Contents

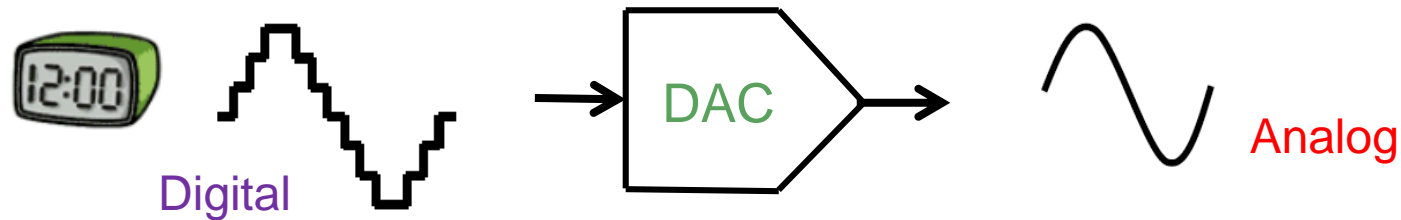
---

- Research Objective
- Segment Type DA Converter
- Characteristic of Variation in Circuit Element
- Proposed Method
  - Magic Square
  - Latin Square
- Conclusion

# Research Objective

## Objective

- Development of a highly linear digital-to-analog converter (DAC)



## Our Approach

- DAC layout technique to cancel systematic mismatch effects among unit current cells.
  - Layout based on **Magic and Latin Squares**

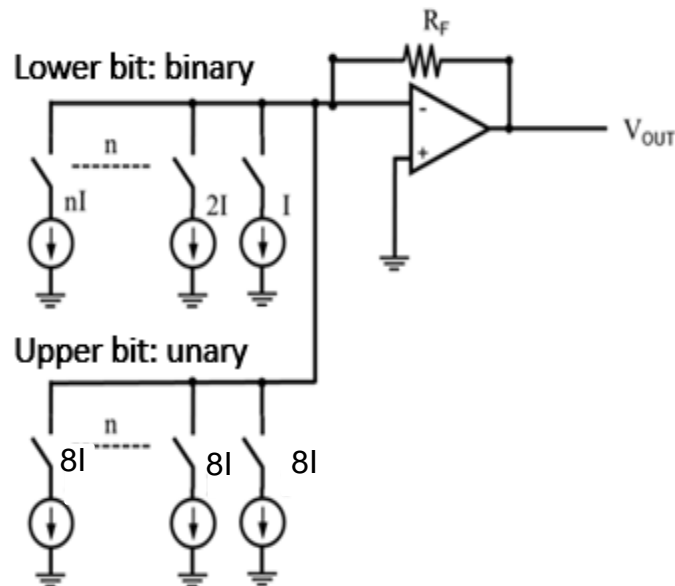
New!!

# Contents

---

- Research Objective
- **Segment Type DA Converter**
- Characteristic of Variation in Circuit Element
- Proposed Method
  - Magic Square
  - Latin Square
- Conclusion

# Segment Type DAC Configuration



## ✓ Binary (Lower bits)

- Small circuit
- Large glitch
- Large mismatch effect & Large nonlinearity

## ✓ Unary (Upper bits)

- Large circuit
- Small glitch
- Small mismatch effect & modest linearity

Segmented DAC

Focus !!

# Segment Type DAC (7-bit case)

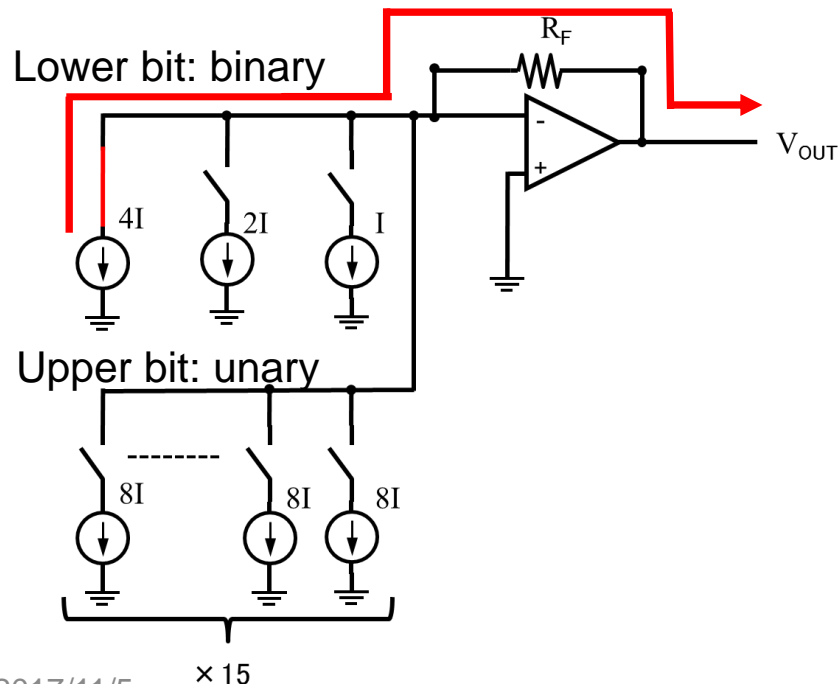
ex.1

In case digital input = 4

(0000100)



$$V_{out} = 4IR_F$$



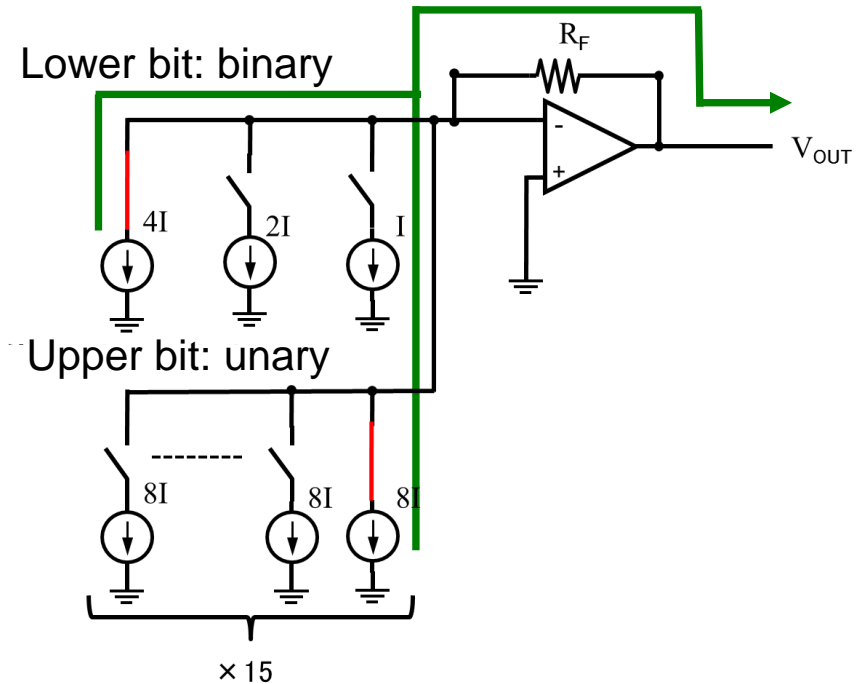
ex.2

In case digital input = 12

(0001100)



$$V_{out} = 12IR_F$$

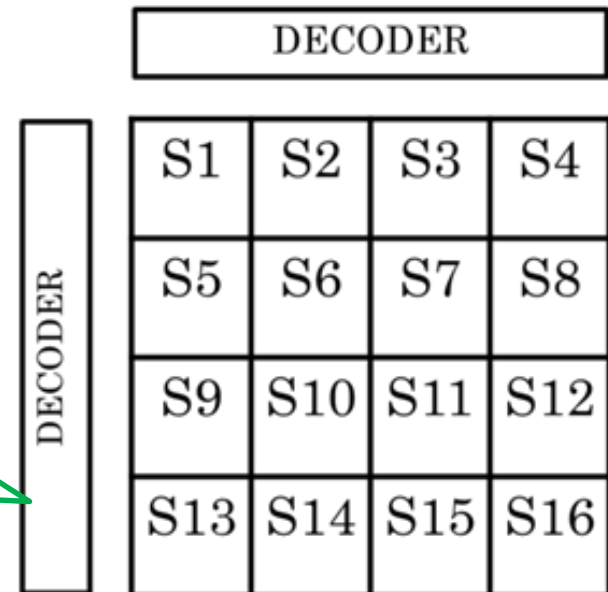
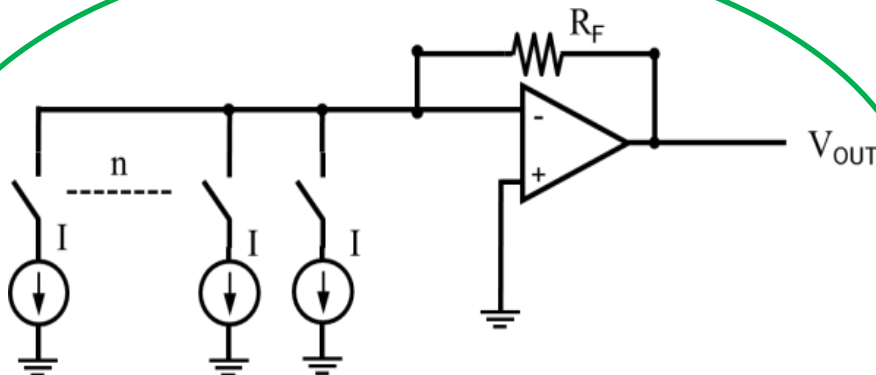


# Unary DAC Features

- Identical current sources
- Small glitch
- Inherent monotonicity



- Large circuits
  - Decoder
  - Many switches and current sources





# Unary DAC Current Cells Layout

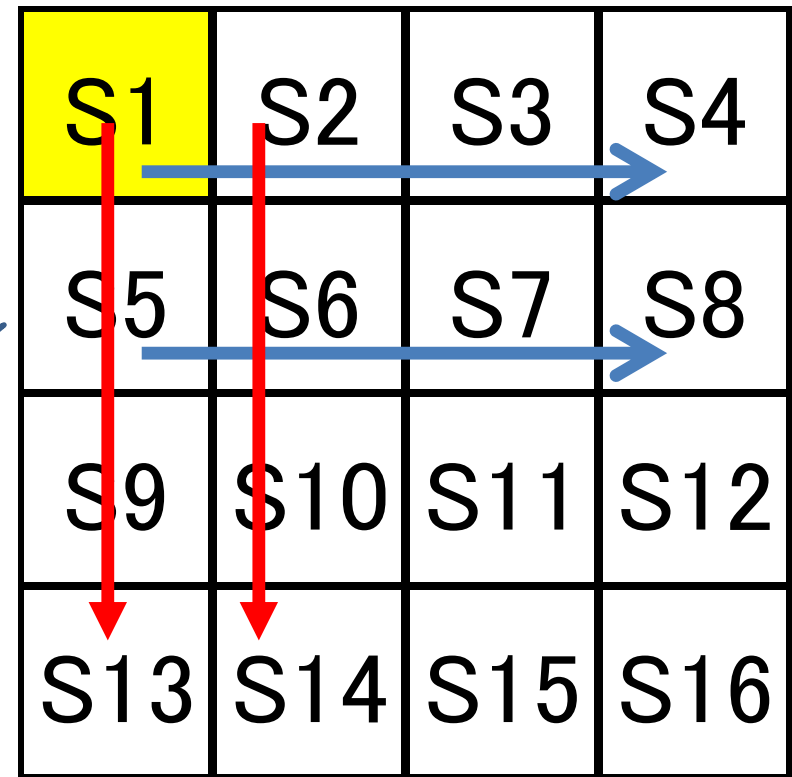
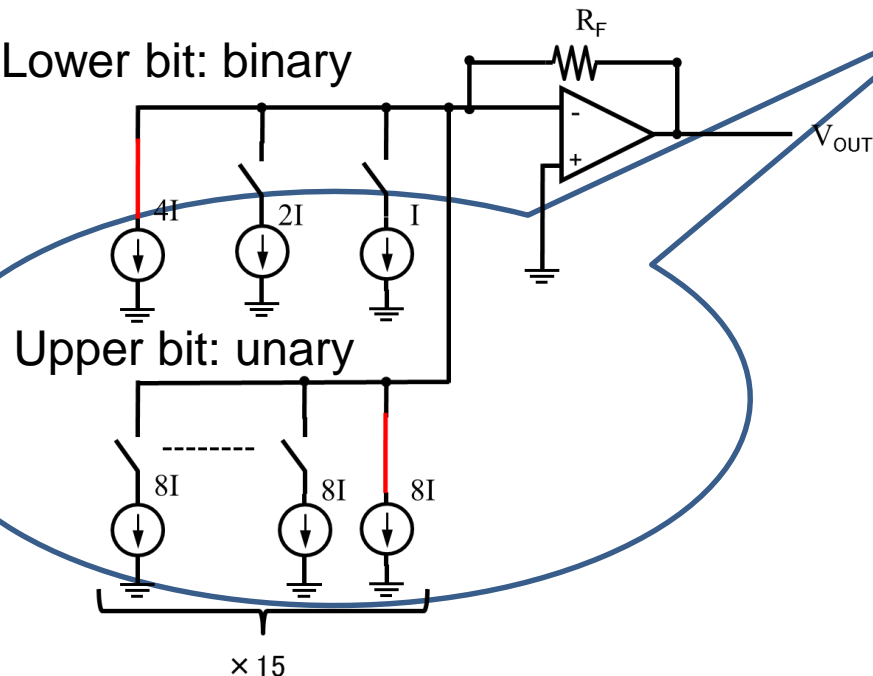
## ✓ 7bit DA Converter

(0001100)



$$V_{out} = 12IR_F$$

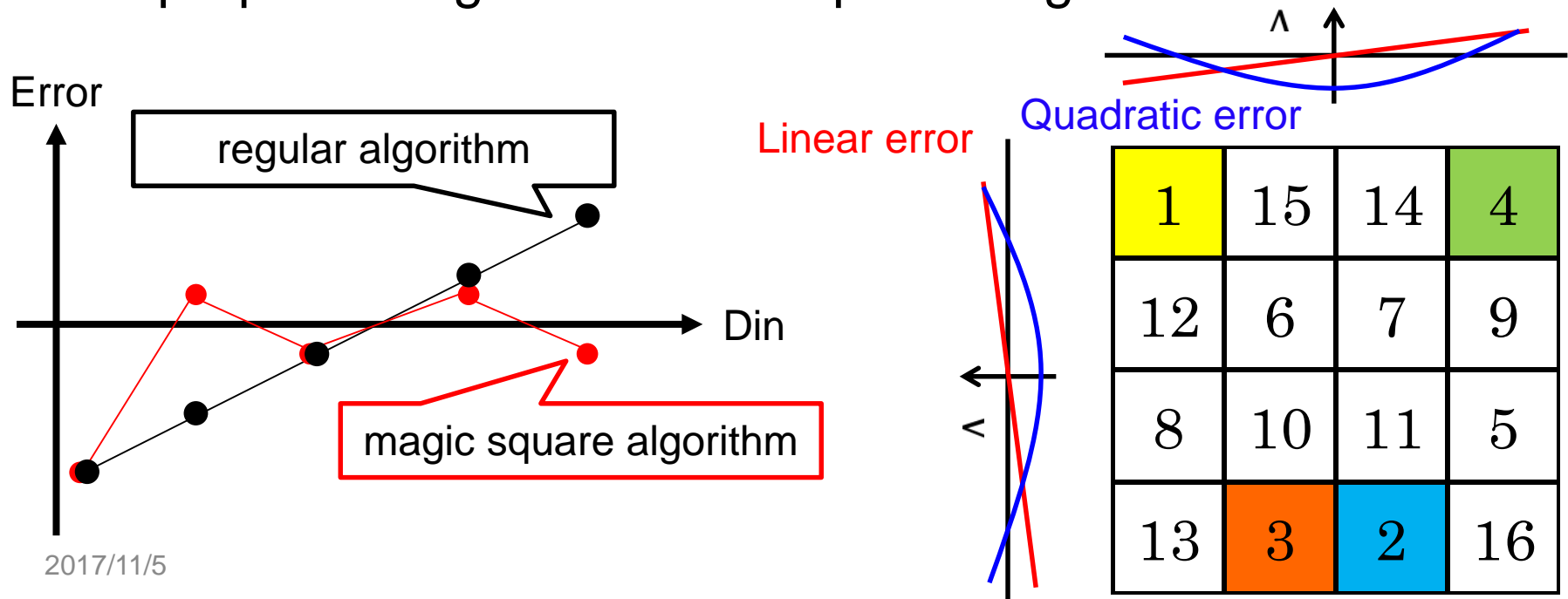
Lower bit: binary



Unit current cell (unary)

# Cell Layout and Systematic Mismatch

- Semiconductor devices have systematic mismatches
- Changing the unit cell layout order
  - ➡ Cancellation of **systematic mismatch effects**
- We propose magic and Latin squares algorithms



# Contents

---

- Research Objective
- Segment Type DA Converter
- **Characteristic of Variation in Circuit Element**
- Proposed Method
  - Magic Square
  - Latin Square
- Conclusion

# Variation in Circuit Element Characteristics

## ◆ Systematic variations

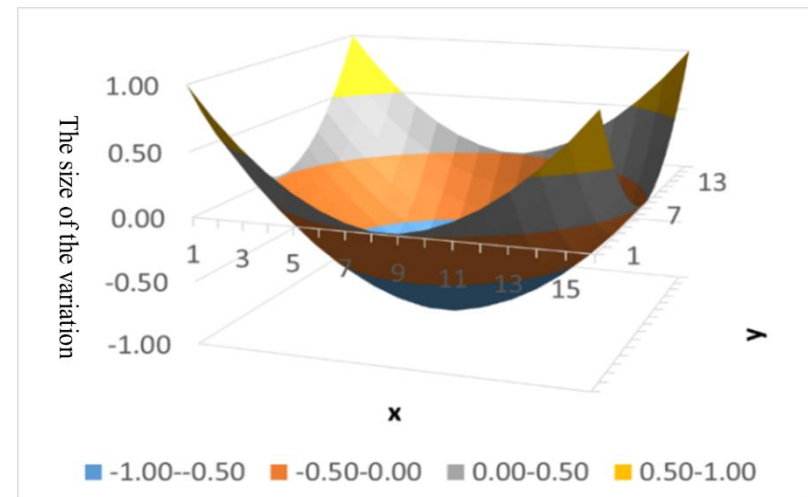
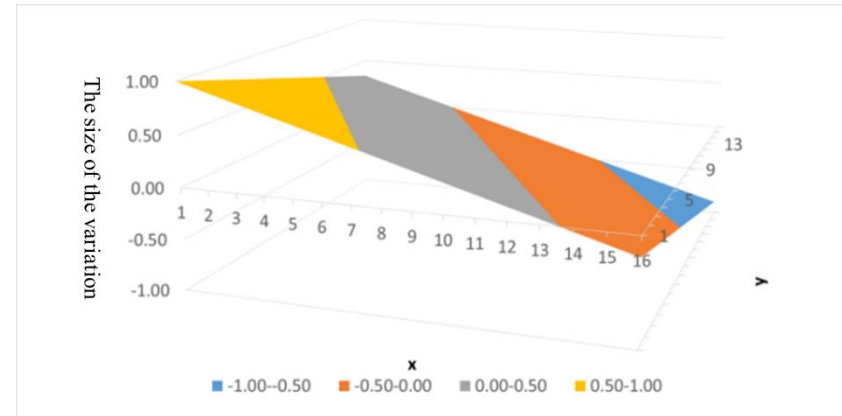
- ✓ Voltage drop
- ✓ Thickness of oxide film
- ✓ Doping
- ✓ Mechanical stress
- ✓ Temperature distribution
- ✓ In wafer plane

Linear  
error

Quadratic  
error



Joint Error (Sum of both)



# Systematic Variation Model

## Linear Error

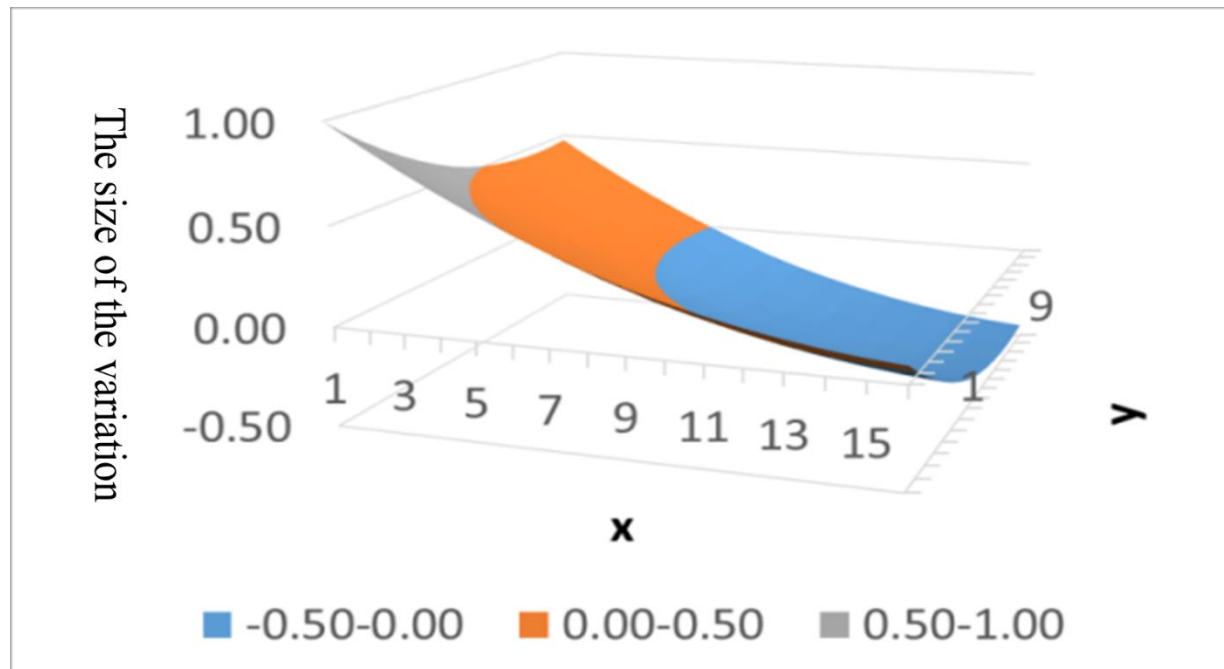
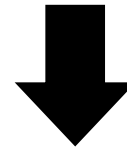
$$\varepsilon_l(x, y) = g_l * \cos \theta * x + g_l * \sin \theta * y$$

## Quadratic Error

$$\varepsilon_q(x, y) = g_q * (x^2 + y^2) - a_0$$

Joint Errors

$$\varepsilon_j(x, y) = \varepsilon_l(x, y) + \varepsilon_q(x, y)$$



# Contents

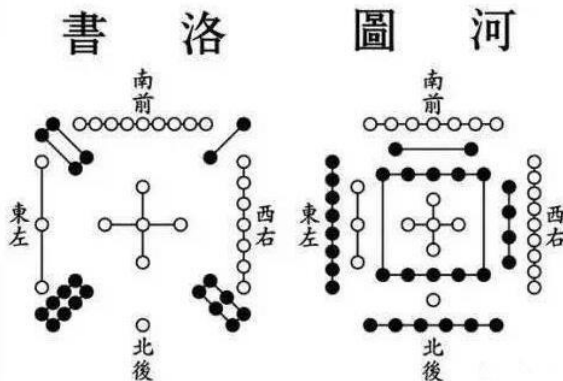
---

- Research Objective
- Segment Type DA Converter
- Characteristic of Variation in Circuit Element
- **Proposed Method**
  - **Magic Square**
  - Latin Square
- Conclusion

# What is Magic Square ?

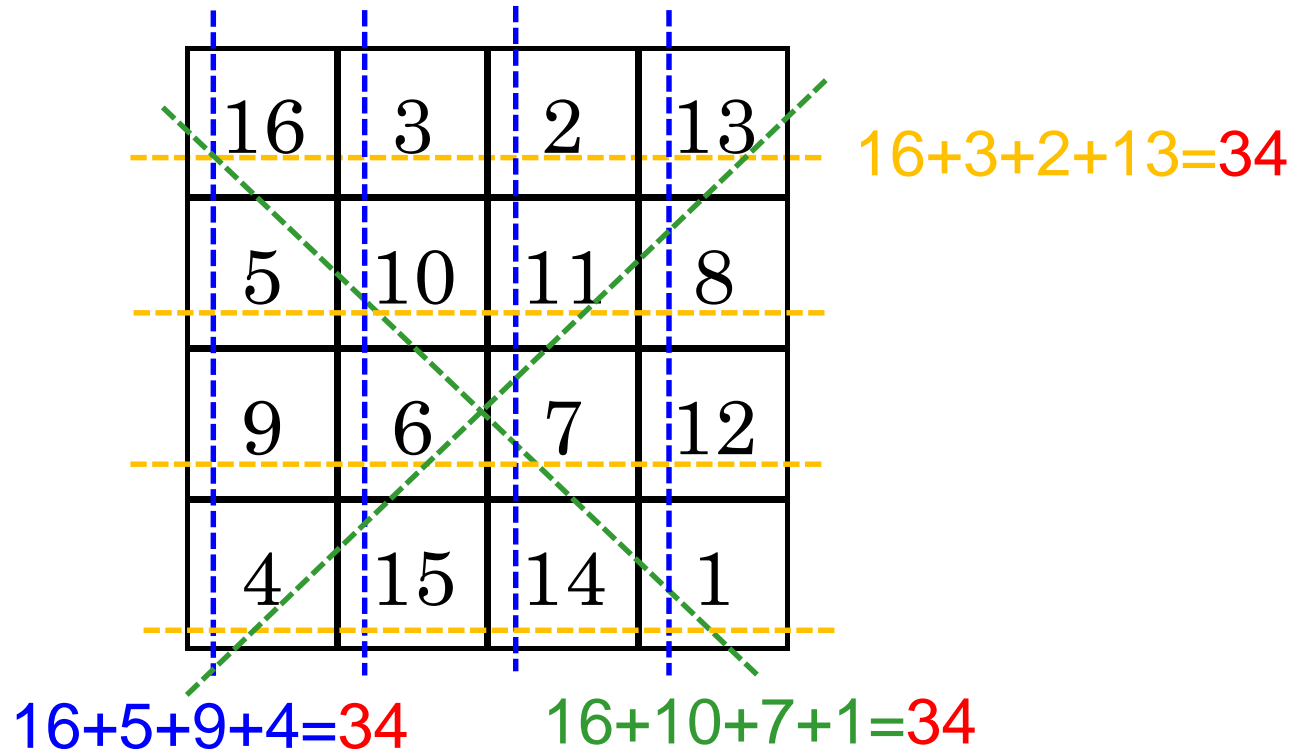
- Classical mathematics
- Origin from Chinese academia
- “Constant sum” characteristics
- Varieties of magic squares

## 3 × 3 Magic Square



2	9	4
7	5	3
6	1	8

# Magic Square Features



Constant Sum  
Row, Column, Diagonal



Magic Square is  
good balance





# Magic Square for Layout Algorithm

## ✓ Concentric Magic Square

Even if one side is removed from the outside, it does not lose compatibility

Numbers are  
in symmetrical positions



expected

Effective for cancellation of  
systematic variation effects

59	5	4	62	63	1	8	58
9	18	17	49	50	42	19	56
55	20	28	33	29	40	45	10
54	44	38	31	35	26	21	11
12	43	39	30	34	27	22	53
13	24	25	36	32	37	41	52
51	46	48	16	15	23	47	14
7	60	61	3	2	64	57	6

# Magic Square for 16x16 Cell Layout

## ◆ Concentric Magic Square

8-bit unit current source cells  
by combining 8-th order squares

59	5	4	62	63	1	8	58
9	18	17	49	50	42	19	56
55	20	28	33	29	40	45	10
54	44	38	31	35	26	21	11
12	43	39	30	34	27	22	53
13	24	25	36	32	37	41	52
51	46	48	16	15	23	47	14
7	60	61	3	2	64	57	6

A1	B1
B2	A2

A: Magic square

B: 45 ° counterclockwise  
rotation

# 16x16 Cell Layout Details

## ◆ Concentric Magic Square

- algorithm

A1	B1
B2	A2

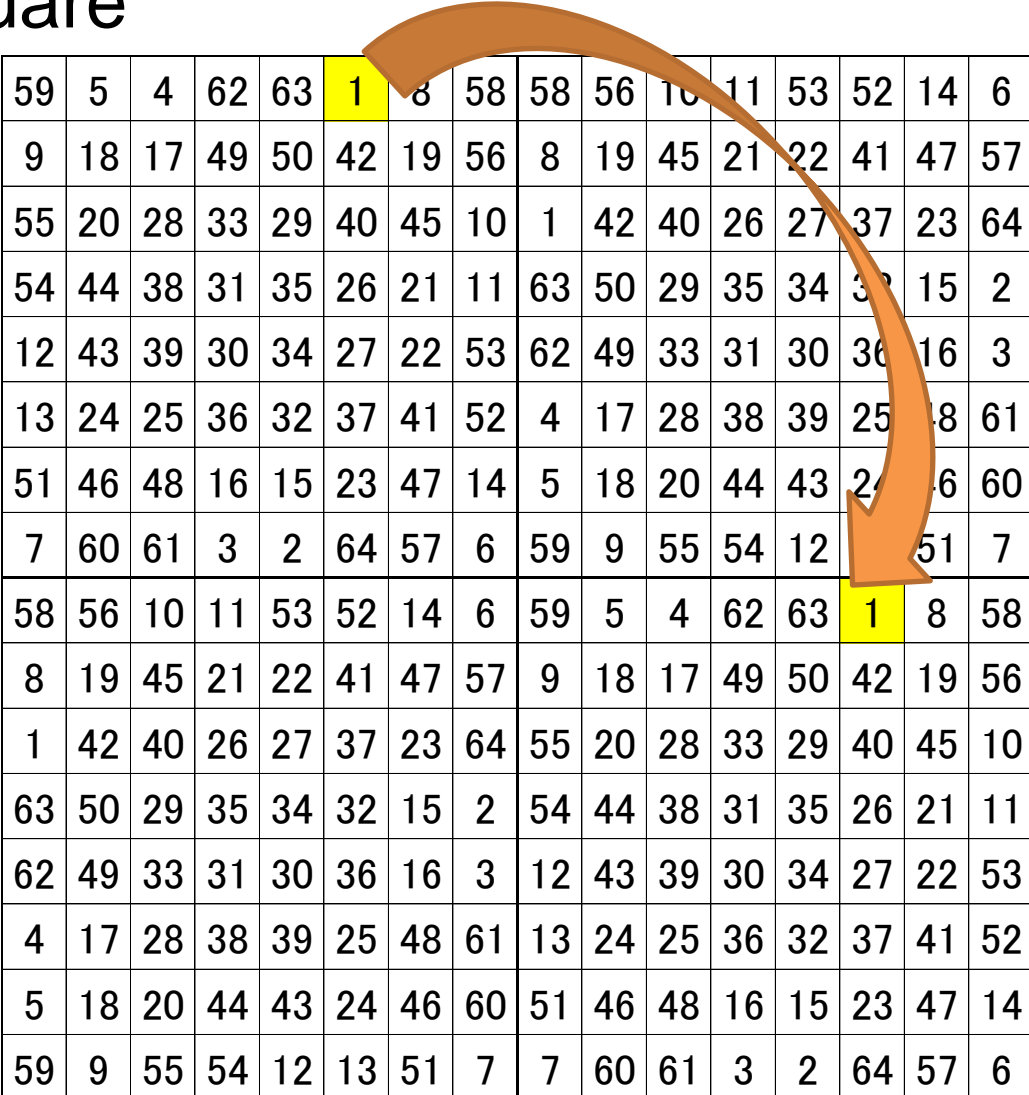
59	5	4	62	63	1	8	58	58	56	10	11	53	52	14	6
9	18	17	49	50	42	19	56	8	19	45	21	22	41	47	57
55	20	28	33	29	40	45	10	1	42	40	26	27	37	23	64
54	44	38	31	35	26	21	11	63	50	29	35	34	32	15	2
12	43	39	30	34	27	22	53	62	49	33	31	30	36	16	3
13	24	25	36	32	37	41	52	4	17	28	38	39	25	48	61
51	46	48	16	15	23	47	14	5	18	20	44	43	24	46	60
7	60	61	3	2	64	57	6	59	9	55	54	12	13	51	7
58	56	10	11	53	52	14	6	59	5	4	62	63	1	8	58
8	19	45	21	22	41	47	57	9	18	17	49	50	42	19	56
1	42	40	26	27	37	23	64	55	20	28	33	29	40	45	10
63	50	29	35	34	32	15	2	54	44	38	31	35	26	21	11
62	49	33	31	30	36	16	3	12	43	39	30	34	27	22	53
4	17	28	38	39	25	48	61	13	24	25	36	32	37	41	52
5	18	20	44	43	24	46	60	51	46	48	16	15	23	47	14
59	9	55	54	12	13	51	7	7	60	61	3	2	64	57	6

# 16x16 Cell Layout Details

## ◆ Concentric Magic Square

- Unit cell selection algorithm

A1	B1
B2	A2



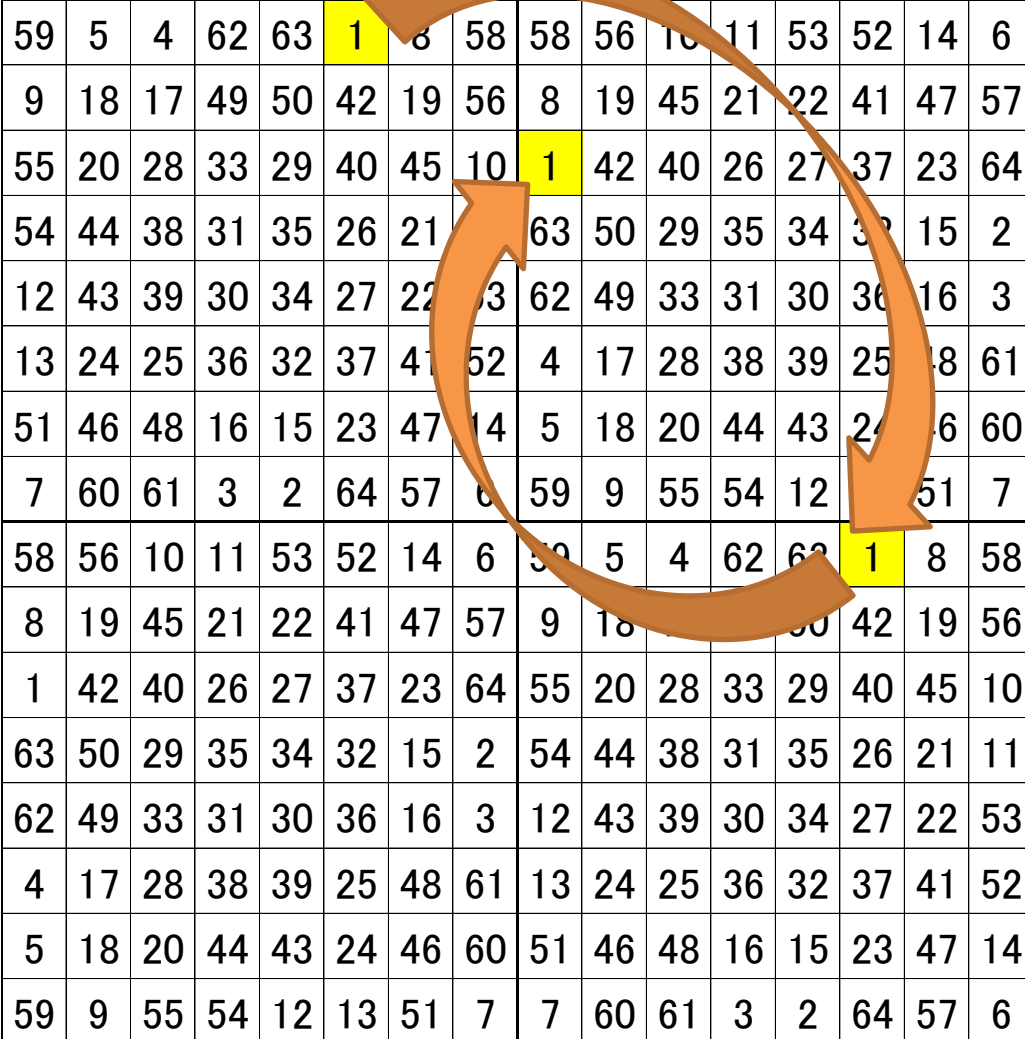
59	5	4	62	63	1	8	58	58	56	10	11	53	52	14	6
9	18	17	49	50	42	19	56	8	19	45	21	22	41	47	57
55	20	28	33	29	40	45	10	1	42	40	26	27	37	23	64
54	44	38	31	35	26	21	11	63	50	29	35	34	32	15	2
12	43	39	30	34	27	22	53	62	49	33	31	30	36	16	3
13	24	25	36	32	37	41	52	4	17	28	38	39	25	48	61
51	46	48	16	15	23	47	14	5	18	20	44	43	24	46	60
7	60	61	3	2	64	57	6	59	9	55	54	12	13	51	7
58	56	10	11	53	52	14	6	59	5	4	62	63	1	8	58
8	19	45	21	22	41	47	57	9	18	17	49	50	42	19	56
1	42	40	26	27	37	23	64	55	20	28	33	29	40	45	10
63	50	29	35	34	32	15	2	54	44	38	31	35	26	21	11
62	49	33	31	30	36	16	3	12	43	39	30	34	27	22	53
4	17	28	38	39	25	48	61	13	24	25	36	32	37	41	52
5	18	20	44	43	24	46	60	51	46	48	16	15	23	47	14
59	9	55	54	12	13	51	7	7	60	61	3	2	64	57	6

# 16x16 Cell Layout Details

## ◆ Concentric Magic Square

- Unit cell selection algorithm

A1	B1
B2	A2



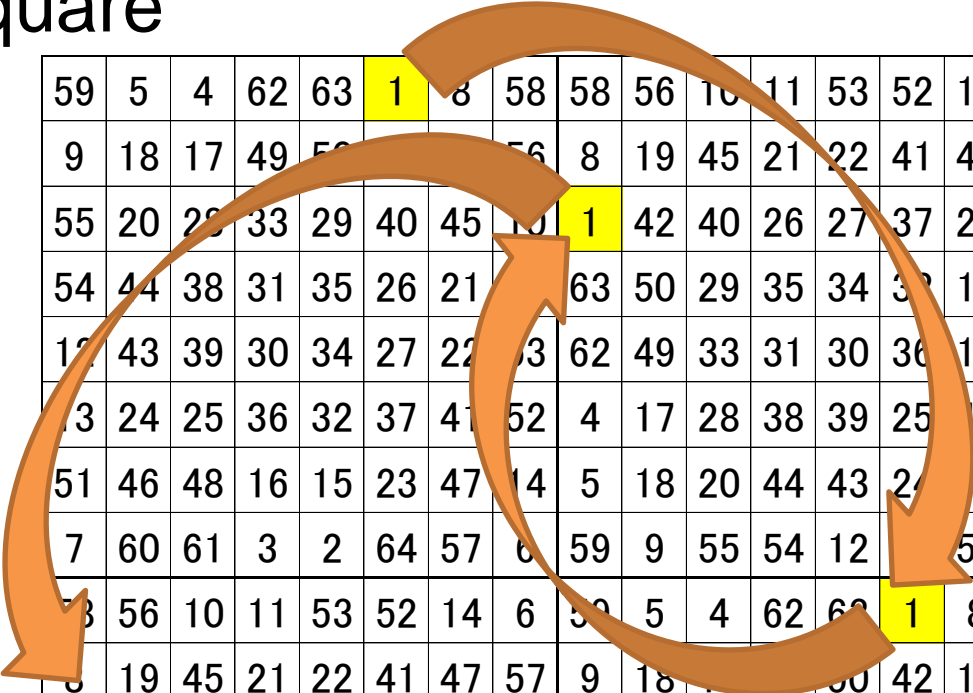
59	5	4	62	63	1	8	58	58	56	10	11	53	52	14	6
9	18	17	49	50	42	19	56	8	19	45	21	22	41	47	57
55	20	28	33	29	40	45	10	1	42	40	26	27	37	23	64
54	44	38	31	35	26	21	63	50	29	35	34	32	15	2	
12	43	39	30	34	27	22	63	62	49	33	31	30	36	16	3
13	24	25	36	32	37	41	52	4	17	28	38	39	25	48	61
51	46	48	16	15	23	47	14	5	18	20	44	43	24	46	60
7	60	61	3	2	64	57	6	59	9	55	54	12	51	7	
58	56	10	11	53	52	14	6	59	5	4	62	63	1	8	58
8	19	45	21	22	41	47	57	9	18	20	44	43	24	46	60
1	42	40	26	27	37	23	64	55	20	28	33	29	40	45	10
63	50	29	35	34	32	15	2	54	44	38	31	35	26	21	11
62	49	33	31	30	36	16	3	12	43	39	30	34	27	22	53
4	17	28	38	39	25	48	61	13	24	25	36	32	37	41	52
5	18	20	44	43	24	46	60	51	46	48	16	15	23	47	14
59	9	55	54	12	13	51	7	7	60	61	3	2	64	57	6

# 16x16 Cell Layout Details

## ◆ Concentric Magic Square

- Unit cell selection algorithm

A1	B1
B2	A2



59	5	4	62	63	1	8	58	58	56	10	11	53	52	14	6
9	18	17	49	50	56	8	19	45	21	22	41	47	57		
55	20	28	33	29	40	45	10	1	42	40	26	27	37	23	64
54	44	38	31	35	26	21		63	50	29	35	34	32	15	2
12	43	39	30	34	27	22	63	62	49	33	31	30	36	16	3
13	24	25	36	32	37	41	52	4	17	28	38	39	25	48	61
51	46	48	16	15	23	47	14	5	18	20	44	43	24	46	60
7	60	61	3	2	64	57	6	59	9	55	54	12		51	7
53	56	10	11	53	52	14	6	59	5	4	62	60	1	8	58
5	19	45	21	22	41	47	57	9	18		30	42	19	56	
1	42	40	26	27	37	23	64	55	20	28	33	29	40	45	10
63	50	29	35	34	32	15	2	54	44	38	31	35	26	21	11
62	49	33	31	30	36	16	3	12	43	39	30	34	27	22	53
4	17	28	38	39	25	48	61	13	24	25	36	32	37	41	52
5	18	20	44	43	24	46	60	51	46	48	16	15	23	47	14
59	9	55	54	12	13	51	7	7	60	61	3	2	64	57	6

# 16x16 Cell Layout Details

## ◆ Concentric Magic Square

- Unit cell selection algorithm

A1	B1
B2	A2

59	5	4	62	63	1	8	58	58	56	10	11	53	52	14	6
9	18	17	49	50			56	8	19	45	21	22	41	47	57
55	20	28	33	29	40	45	10	1	42	40	26	27	37	23	64
54	44	38	31	35	26	21		63	50	29	35	34	32	15	2
12	43	39	30	34	27	22	13	62	49	33	31	30	36	16	3
13	24	25	36	32	37	41	52	4	17	28	38	39	25	18	61
51	46	48	16	15	23	47	14	5	18	20	44	43	24	16	60
7	60	61	3	2	64	57	6	59	9	55	54	12		51	7
13	56	10	11		52	14	6	50	5	4	62	63	1	8	58
19	45	21	22		1	47	57	9	18		50	42	19	56	
1	4	40	26	27	7	23	64	55	20	28	33	29	40	45	10
63	50		25	24	32	15	2	54	44	38	31	35	26	21	11
62	49	33	31	30	36	16	3	12	43	39	30	34	27	22	53
4	17	28	38	39	25	48	61	13	24	25	36	32	37	41	52
5	18	20	44	43	24	46	60	51	46	48	16	15	23	47	14
59	9	55	54	12	13	51	7	7	60	61	3	2	64	57	6

# 16x16 Cell Layout Details

## ◆ Concentric Magic Square

- Unit cell selection algorithm

A1	B1
B2	A2

59	5	4	62	63	1	8	58	58	56	10	11	53	52	14	6
9	18	17	49	50	42	19	56	8	19	45	21	22	41	47	57
55	20	28	33	29	40	45	10	1	42	40	26	27	37	23	64
54	44	38	31	35	26	21	11	63	50	29	35	34	32	15	2
12	43	39	30	34	27	22	53	62	49	33	31	30	36	16	3
13	24	25	36	32	37	41	52	4	17	28	38	39	25	48	61
51	46	48	16	15	23	47	14	5	18	20	44	43	24	46	60
7	60	61	3	2	64	57	6	59	9	55	54	12	13	51	7
58	56	10	11	53	52	14	6	59	5	4	62	63	1	8	58
8	19	45	21	22	41	47	57	9	18	17	49	50	42	19	56
1	42	40	26	27	37	23	64	55	20	28	33	29	40	45	10
63	50	29	35	34	32	15	2	54	44	38	31	35	26	21	11
62	49	33	31	30	36	16	3	12	43	39	30	34	27	22	53
4	17	28	38	39	25	48	61	13	24	25	36	32	37	41	52
5	18	20	44	43	24	46	60	51	46	48	16	15	23	47	14
59	9	55	54	12	13	51	7	7	60	61	3	2	64	57	6



# 16x16 Cell Layout Details

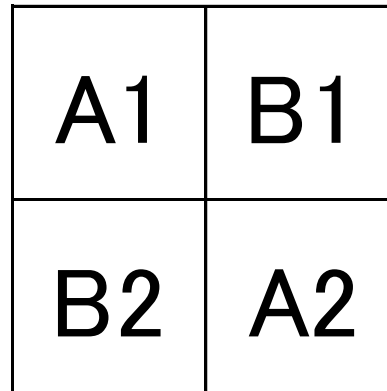
## ◆ Concentric Magic Square

### ● algorithm

1. 1 in A1
2. 1 in A2
3. 1 in B1
4. 1 in B2
5. 2 in A1
- ⋮

1023. 256 in B1

1024. 256 in B2



59	5	4	62	63	1	8	58	58	56	10	11	53	52	14	6
9	18	17	49	50	42	19	56	8	19	45	21	22	41	47	57
55	20	28	33	29	40	45	10	1	42	40	26	27	37	23	64
54	44	38	31	35	26	21	11	63	50	29	35	34	32	15	2
12	43	39	30	34	27	22	53	62	49	33	31	30	36	16	3
13	24	25	36	32	37	41	52	4	17	28	38	39	25	48	61
51	46	48	16	15	23	47	14	5	18	20	44	43	24	46	60
7	60	61	3	2	64	57	6	59	9	55	54	12	13	51	7
58	56	10	11	53	52	14	6	59	5	4	62	63	1	8	58
8	19	45	21	22	41	47	57	9	18	17	49	50	42	19	56
1	42	40	26	27	37	23	64	55	20	28	33	29	40	45	10
63	50	29	35	34	32	15	2	54	44	38	31	35	26	21	11
62	49	33	31	30	36	16	3	12	43	39	30	34	27	22	53
4	17	28	38	39	25	48	61	13	24	25	36	32	37	41	52
5	18	20	44	43	24	46	60	51	46	48	16	15	23	47	14
59	9	55	54	12	13	51	7	7	60	61	3	2	64	57	6

Represent pseudo-random switching  
while taking care of center and corners

# Simulation Results (linear error case)

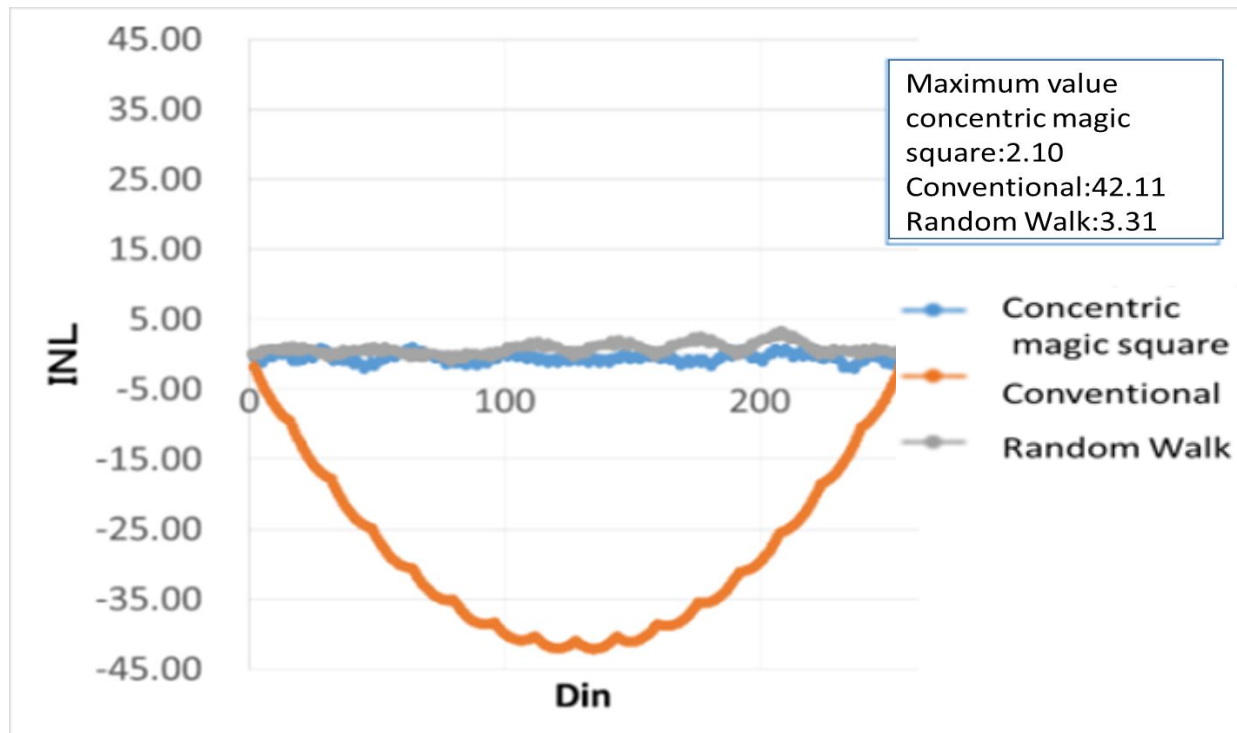
## ◆ Concentric Magic Square

✓ **Linear** Error (Current Cell Systematic Mismatch)

$$\varepsilon_l(x, y) = g_l * \cos \theta * x + g_l * \sin \theta * y$$

$$\theta = 30^\circ$$

$$g_l = 1$$



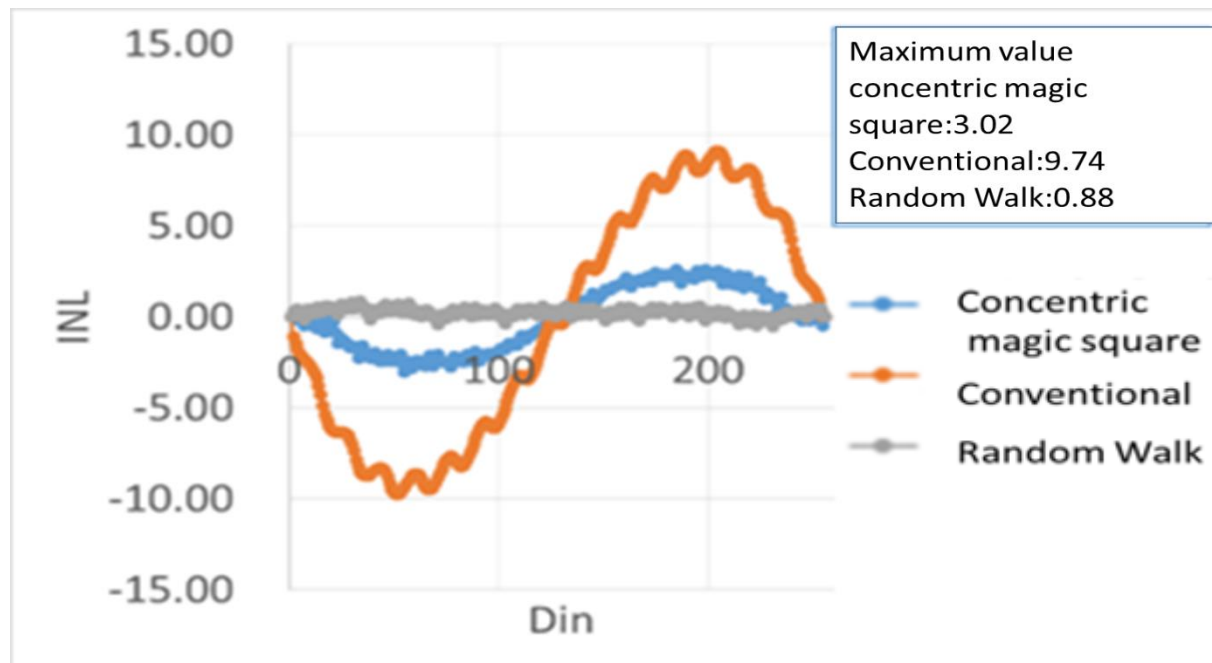
# Simulation Results (quadratic error case)<sup>27/42</sup>

## ◆ Concentric Magic Square

### ✓ Quadratic Error (Current Cell Systematic Mismatch)

$$\varepsilon_q(x, y) = g_q * (x^2 + y^2) - a_0$$

$$g_q = 1, a_0 = 0$$

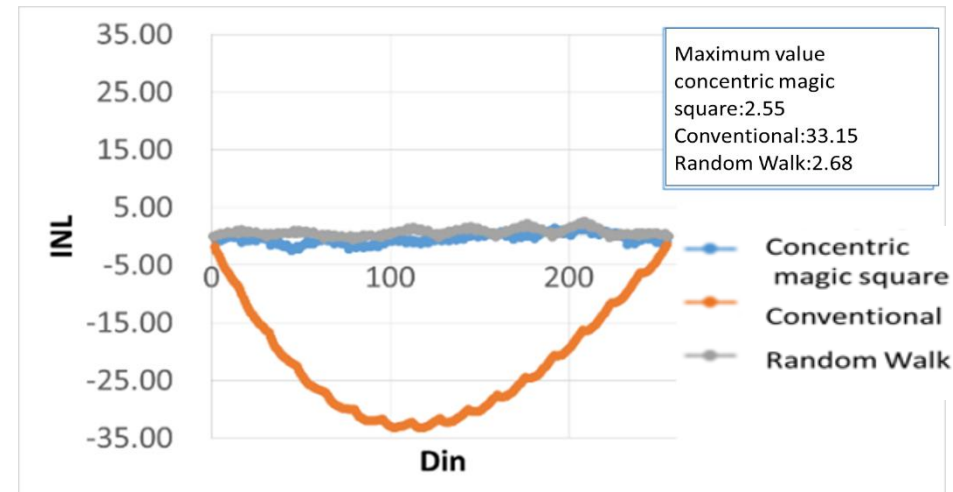
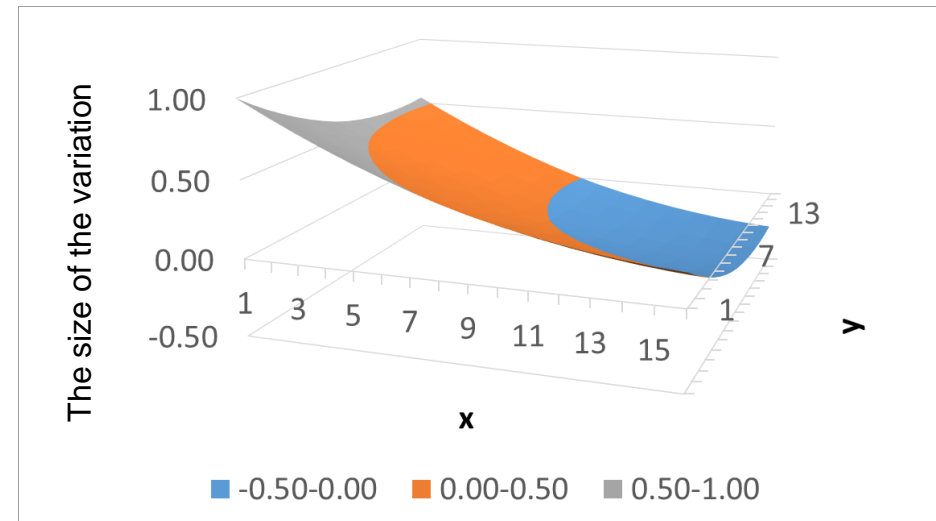


# Simulation Results (joint error case)

## ◆ Concentric Magic Square

✓ Joint Error

Linear > Quadratic case

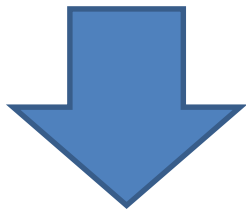


# Simulation Results (joint error case)

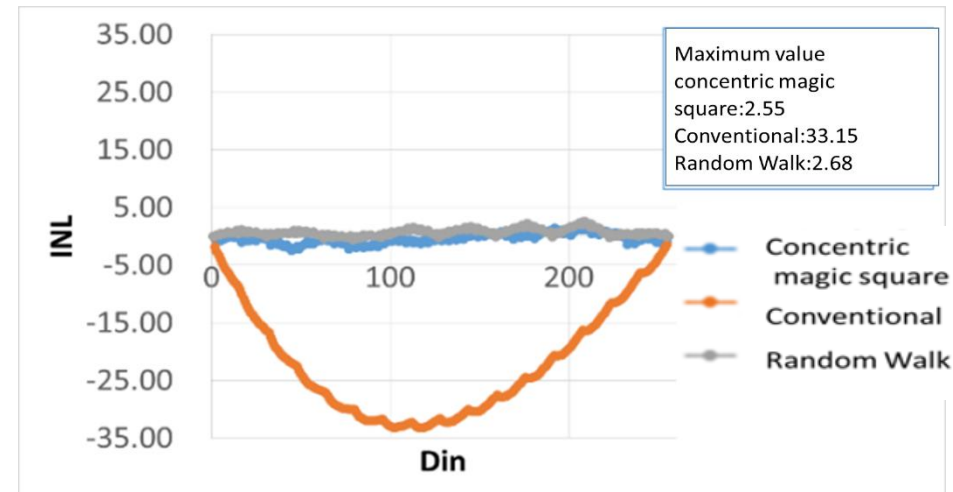
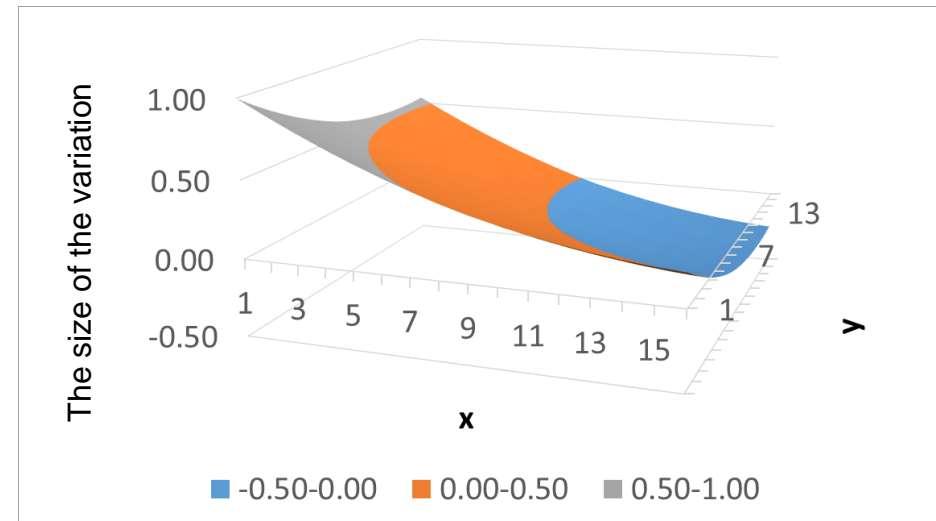
## ◆ Concentric Magic Square

✓ Joint Error

Linear > Quadratic case



Magic square is better

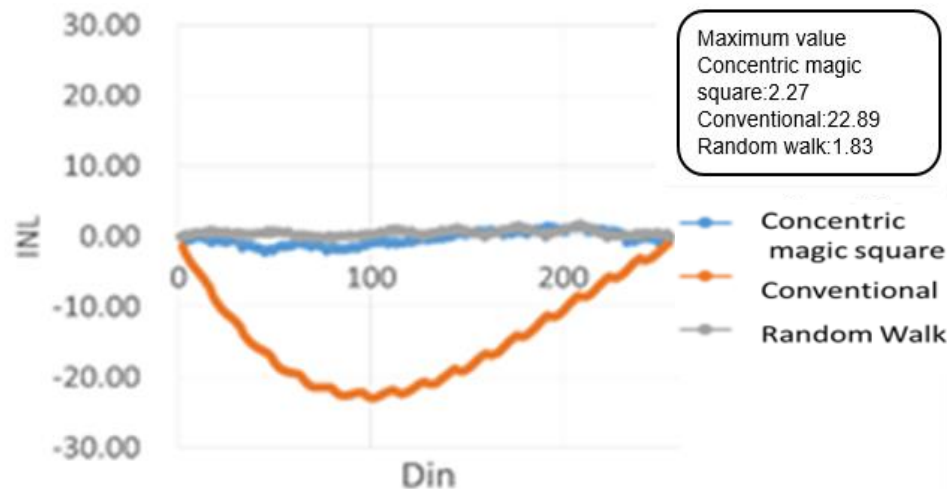
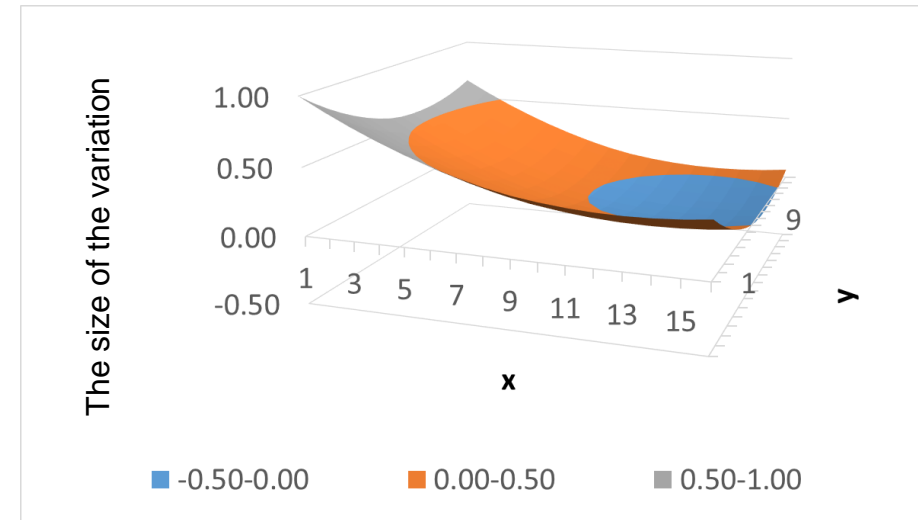


# Simulation Results (joint error case)

## ◆ Concentric Magic Square

✓ Joint Error

Linear < Quadratic case

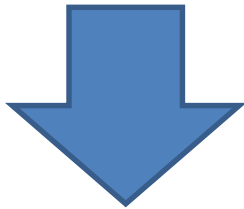


# Simulation Results (joint error case)

## ◆ Concentric Magic Square

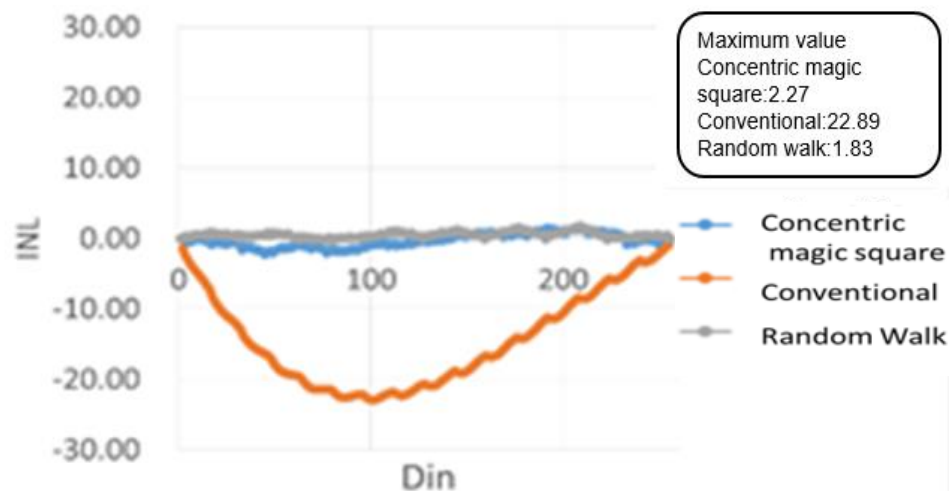
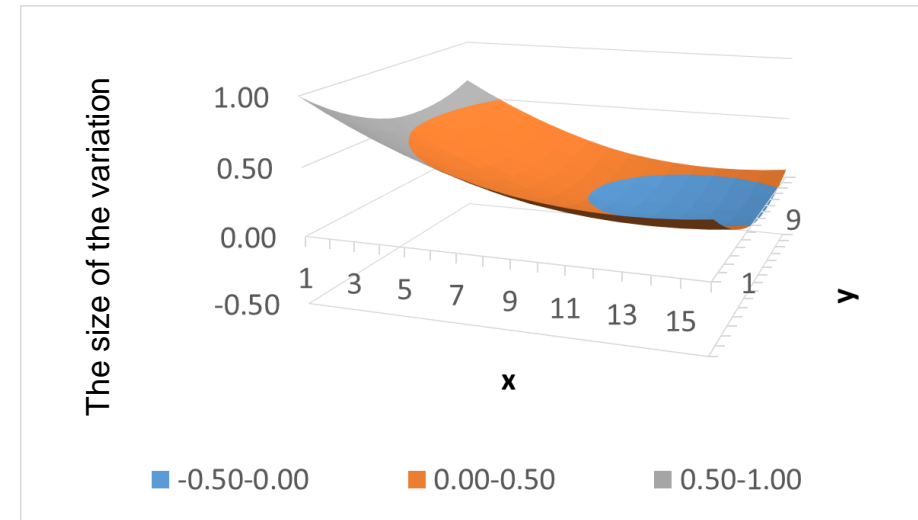
✓ Joint Error

Linear < Quadratic case



Random Walk is better

Is the magic square suitable for temporary variation?



# Contents

---

- Research Objective
- Segment Type DA Converter
- Characteristic of Variation in Circuit Element
- **Proposed Method**
  - Magic Square
  - **Latin Square**
- Conclusion



# What is Latin Square ?

- $n \times n$  array filled with  $n$  different symbols
- Each symbols occurring exactly once in each row and column



Example :

A	B	C
C	A	B
B	C	A

$3 \times 3$  Latin square

1	2	3	4
3	4	1	2
4	3	2	1
2	1	4	3

$4 \times 4$  Latin square

Leonhard Euler(1707-1783)  
Swiss mathematician, physicist

# Latin Square for Layout Algorithm

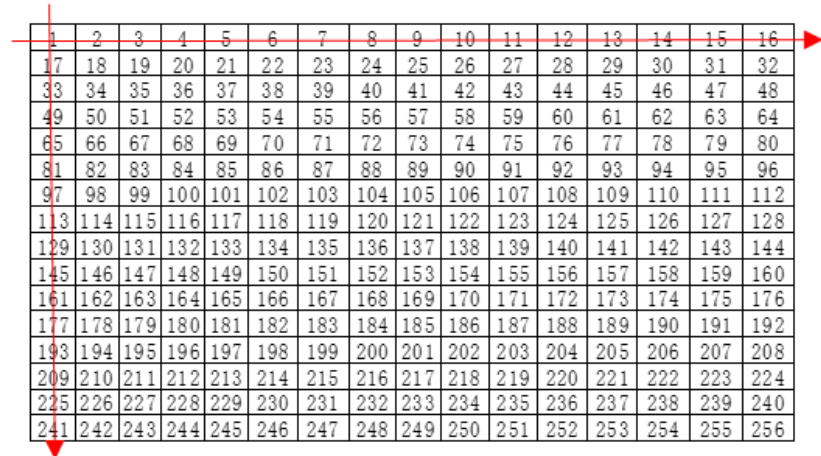
1	2	N	3	N-1	4	n-2	5	n-3	6	n-4	7	n-5	8	n-6	9
---	---	---	---	-----	---	-----	---	-----	---	-----	---	-----	---	-----	---

1	2	16	3	15	4	14	5	13	6	12	7	11	8	10	9
2	3	1	4	16	5	15	6	14	7	13	8	12	9	11	10
3	4	2	5	1	6	16	7	15	8	14	9	13	10	12	11
4	5	3	6	2	7	1	8	16	9	15	10	14	11	13	12
5	6	4	7	3	8	2	9	1	10	16	11	15	12	14	13
6	7	5	8	4	9	3	10	2	11	1	12	16	13	15	14
7	8	6	9	5	10	4	11	3	12	2	13	1	14	16	15
8	9	7	10	6	11	5	12	4	13	3	14	2	15	1	16
9	10	8	11	7	12	6	13	5	14	4	15	3	16	2	1
10	11	9	12	8	13	7	14	6	15	5	16	4	1	3	2
11	12	10	13	9	14	8	15	7	16	6	1	5	2	4	3
12	13	11	14	10	15	9	16	8	1	7	2	6	3	5	4
13	14	12	15	11	16	10	1	9	2	8	3	7	4	6	5
14	15	13	16	12	1	11	2	10	3	9	4	8	5	7	6
15	16	14	1	13	2	12	3	11	4	10	5	9	6	8	7
16	1	15	2	14	3	13	4	12	5	11	6	10	7	9	8

Considering a “complete Latin square”; for even  $n$  ,  
 put the numbers 1 through  $n$  in the first row in the following order :  
 1, 2,  $n$ , 3,  $n-1$ , ...,  $n/2+2$ ,  $n/2+1$ .

# Simulation Conditions

- 8-bit unary DAC
  - Static performance (INL)
  - Dynamic performance (SFDR)



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	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256

- Compared three methods
  - Complete Latin Square
  - Common Centroid
  - Unary Layout



1	3	6	7	15	9	12	13	4	5	8	2	10	11	14	16
3	2	4	5	8	16	10	11	6	7	1	9	12	13	15	14
6	4	2	3	5	7	16	9	8	1	10	12	14	15	13	11
7	5	3	1	4	6	8	15	2	9	11	13	16	14	12	10
15	8	5	4	1	3	6	7	10	11	14	16	13	12	9	2
9	16	7	6	3	2	4	5	12	13	15	14	11	10	1	8
12	10	16	8	6	4	2	3	14	15	13	11	9	1	7	5
13	11	9	15	7	5	3	1	16	14	12	10	2	8	6	4
4	6	8	2	10	12	14	16	1	3	5	7	15	9	11	13
5	7	1	9	11	13	15	14	3	2	4	6	8	16	10	12
8	1	10	11	14	15	13	12	5	4	2	3	6	7	16	9
2	9	12	13	16	14	11	10	7	6	3	1	4	5	8	15
10	12	14	16	13	11	9	2	15	8	6	4	1	3	5	7
11	13	15	14	12	10	1	8	9	16	7	5	3	2	4	6
14	15	13	12	9	1	7	6	11	10	16	8	5	4	2	3
16	14	11	10	2	8	5	4	13	12	9	15	7	6	3	1

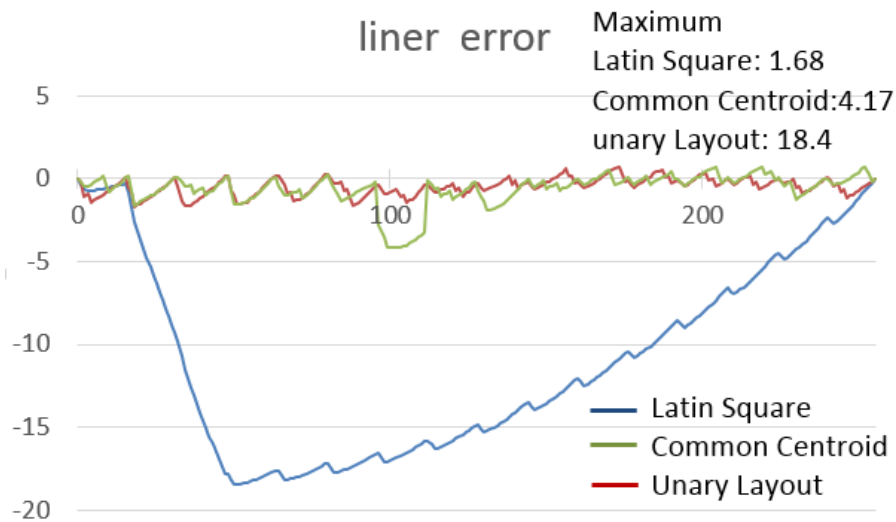
- Mismatch of current sources
  - Current sources have average value of 1.0
  - Random number between  $-1 < \text{mismatch} < +1$  (uniform distribution)

# Simulation Results (INL)

## ◆ Standard Latin square layout algorithm

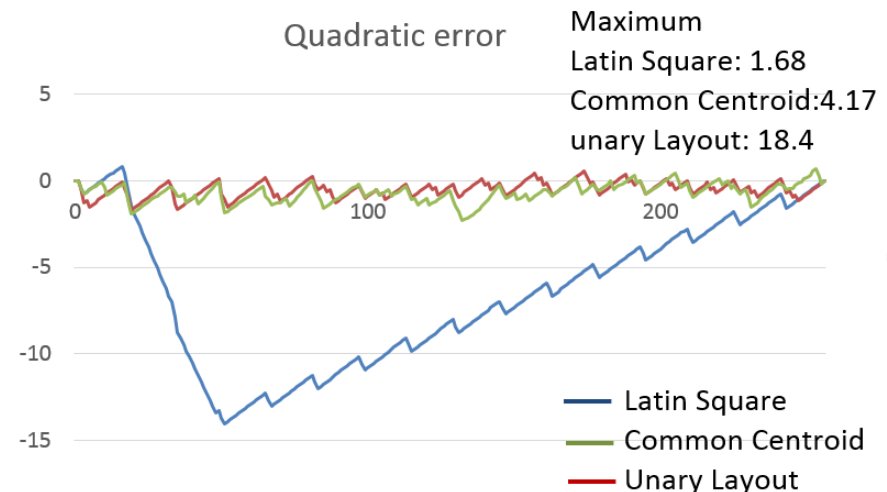
### Linear Error

linear error

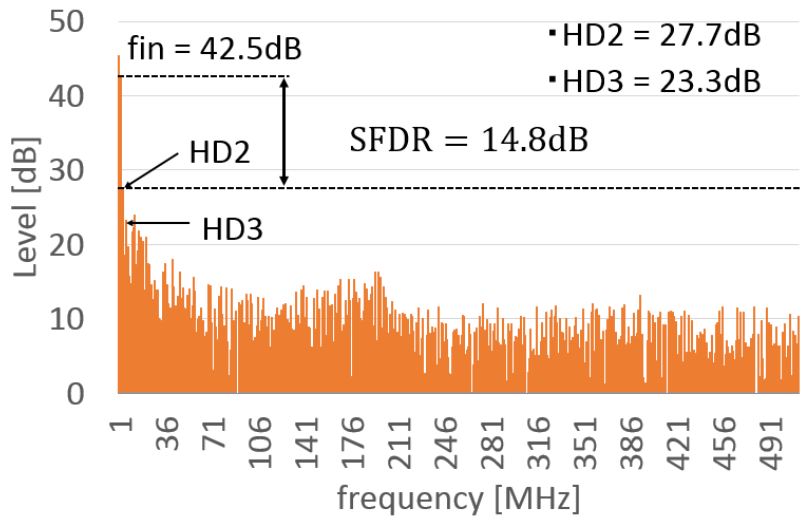


### Quadratic Error

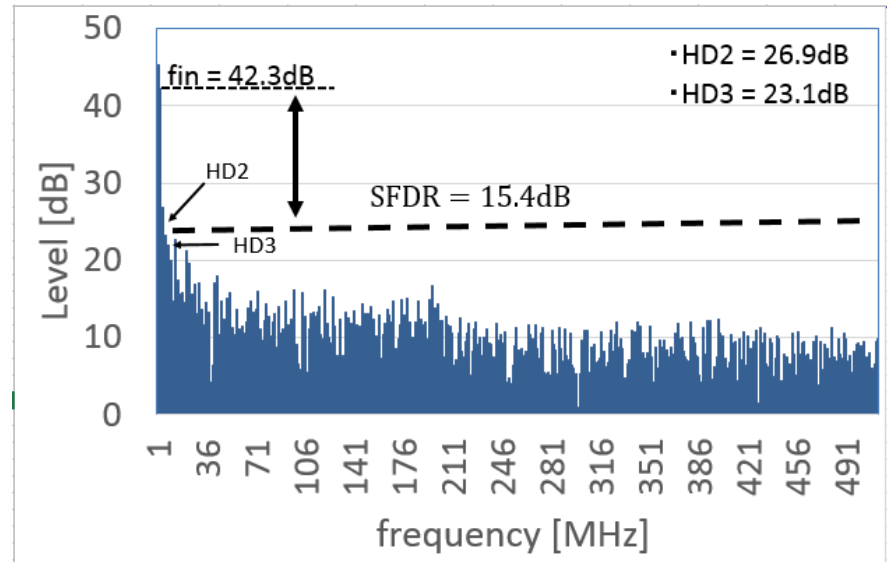
Quadratic error



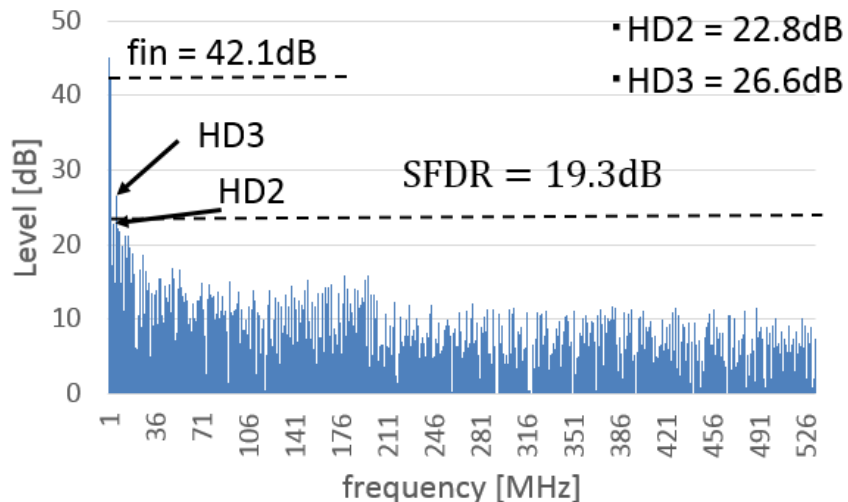
# Simulation Results (SFDR)



Regular layout



Common centroid



SFDR improved !

Latin square

# Summary

---

- Research Objective
- Segment Type DA Converter
- Characteristic of Variation in Circuit Element
- Proposed Method
  - Magic Square
  - Latin Square
- Conclusion

# Conclusion

---

- Unary DAC linearity improvement
  - Unit current cell systematic mismatch effects cancellation
  - Unit current cell layout algorithm based on magic square and Latin square
- Simulation validation
  - INL improvement
  - SFDR improvement

# Final Statement

## 温故知新

Classical mathematics can contribute modern technology.



Leonhard Euler (1707~1783)







# Thank you for listening

## 謝謝



# Digital-to-Analog Converter Layout Technique and Unit Cell Sorting Algorithm for Linearity Improvement Based on Magic Square

Masashi Higashino <sup>1, a</sup>, Shaiful Nizam Bin Mohyar <sup>2, b</sup>, Yao Dan <sup>1, c</sup>

Yifei Sun <sup>1, d</sup>, Anna Kuwana <sup>1, e</sup>, Haruo Kobayashi <sup>1, f</sup>

<sup>1</sup>Division of Electronics and Informatics, Faculty of Science and Technology

Gunma University, Kiryu 376-8515 Japan

<sup>2</sup>School of Microelectronic Eng., University Malaysia Perlis, Pauh Putra Campus,

Arau, Perlis, 02600 Malaysia

<sup>a</sup><t15804080@gunma-u.ac.jp>, <sup>b</sup><nizammohyar@unimap.edu.mhi>, <sup>c</sup><t171d087@gunma-u.ac.jp>

<sup>d</sup><t1610e01@gunma-u.ac.jp>, <sup>e</sup><kuwana.anna@gunma-u.ac.jp>, <sup>f</sup><koba@gunma-u.ac.jp>

**Keywords:** digital-to-analog converter, magic square, linearity, layout, calibration algorithm

**Abstract.** This paper describes techniques to improve the linearity of the unary digital-to-analog converter (DAC). We have developed two techniques based on magic square properties; a layout technique and a switching algorithm of unit current (capacitor) cells to canceling their systematic and/or random mismatch effects. Simulation results and discussions are provided for DAC linearity comparison in cases that the proposed magic square and conventional algorithms are used. We show that there are possibilities to obtain better unary DAC linearity by employing magic square algorithms, depending on mismatch characteristics, because the magic square has well-balanced 2-dimensional characteristics. There are rich mathematical research assets for magic squares and their usage would lead to linearity improvement algorithms for the DAC in nano-CMOS era.

## 1. Introduction

Recently, electronic devices such as mobile phones, wireless modems and tablet terminators demand for small-sized, high-speed and high-linearity digital-to-analog-converters (DACs). Especially, in fields of wireless communication, the accuracy (linearity) of the DAC is very important to ensure the avoidance from the interference leakage during transmission. Semiconductor devices on silicon wafer suffer from random and systematic mismatches regarding to characteristics of MOSFETs, resistors, and capacitors, which cause input and output relationships of the DAC to be non-linear.

In this paper, we have developed two techniques for the unary DAC linearity improvement by using “constant sum” characteristic of the magic square. One is a layout algorithm, while the other is a sorting algorithm or a switching selection algorithm of the unit current (capacitor) cells.

As the first method, we propose a magic square layout technique to improve the linearity of the unary DAC to cancel systematic mismatch effects among unit current (or capacitor) cells. (Precisely speaking, this technique is a unit cell selection algorithm based on magic square, considering the unit cell array layout. For simplicity, we call this technique as a layout technique based on magic square.) The magic square is a kind of a classical mathematics and has well-balanced characteristics [1]. Simulation results and discussions are shown for comparison between magic square and regular layout techniques.

As the second method, we propose a unit cell sorting algorithm based on magic square properties for improvement of the DAC linearity (we consider here both integral nonlinearity (INL) and differential nonlinearity (DNL)). Since not only the static performance but also the dynamic

performance are important for the DAC in many applications, we aim to improve spurious free dynamic range (SFDR). In most cases, the unary DAC decodes a binary code of the digital input into a thermometer code and turns on current source cells depending on it to produce an analog output signal. However in actual chip implementation, the DAC linearity is degraded due to unit current source mismatches. Hence, we propose here a unit cell sorting algorithm based on magic square calibration to cancel the random and systematic mismatch effects. We show its simulation results to demonstrate its effectiveness, and provide some considerations.

Notice that most of high-resolution DACs consist of unary configuration for higher bits and binary one for lower bits due to good balance of performance, chip area and power. The unary configuration part for higher bits is important for the overall DAC linearity [3]. Then we focus on the unary part here.

## 2. Problem Formulation

### 2.1 Unary-DAC Architecture

A unary DAC employs a unary weighted (identical) current source structure (Fig.1). The unary weighted structure requires  $2^N-1$  unit current sources for N-bit resolution where all current source weights are identical as expressed in eq. (1):

$$I_1 = I_2 = \dots = I_{14} = I_{15} \quad (1)$$

For the operation of the unary weighted structure, a binary-to-thermometer code decoder is required. The unary DAC has characteristics of small glitch and inherent monotonicity, even though this architecture suffers from large decoder circuits and many switches which increase silicon chip area.

### 2.2 DAC Nonlinearity

In practical CMOS technologies, the current source mismatches are influenced by their threshold voltage ( $V_{th}$ ) mismatch and/or by the slope ( $\beta$ ) mismatch (Fig.2) [2]. The nominal drain current ( $I_d$ ) in saturation region is given by:

$$I_d = \frac{\beta}{2} (|V_{gs}| - |V_{th}|)^2$$

Also drain current mismatch is given by:

$$\frac{\Delta I_d}{I_d} = \frac{2}{|V_{gs}| - |V_{th}|} \frac{A_{v_{th}} t_{ox}}{\sqrt{WL}}$$

Here  $I_d$  is its drain current,  $\Delta I_d$  is its mismatch (standard deviation),  $\beta$  is its current slope,  $V_{gs}$  is its gate-source voltage,  $V_{th}$  is its gate-source voltage,  $W$  is its channel width,  $L$  is its channel length,  $t_{ox}$  is its gate oxide thickness,  $A_{v_{th}}$  is a constant (extracted parameter value). We see that current mismatches are dependent on their device sizes ( $W, L$ ). Note that here, we are considering to reduce the DAC nonlinearity effects of the current mismatches due to small device size  $(\sqrt{WL})$ .

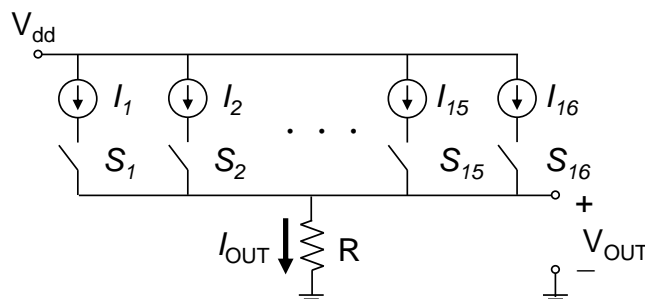


Fig.1. Unary digital-to-analog converter.

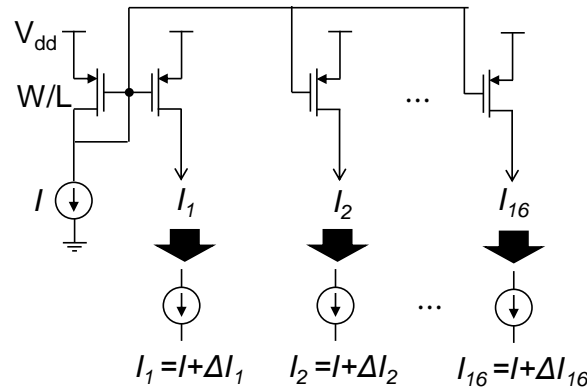


Fig. 2.  $\Delta I_1$ ,  $\Delta I_2$  are current mismatches for  $I_1$ ,  $I_2$ , respectively.

When we consider the current mismatches in Fig. 2, eq. (1) is changed to as follows:

$$I_1 = I + \Delta I_1, \quad I_2 = I + \Delta I_2, \quad \dots, \quad I_{15} = I + \Delta I_{15}$$

If we define  $I$  as follows:

$$I = (I_1 + I_2 + \dots + I_{14} + I_{15})/15$$

then we have the following:

$$\Delta I_1 + \Delta I_2 + \dots + \Delta I_{14} + \Delta I_{15} = 0.$$

### 2.3 Magic Square

Characteristic of the magic square is a constant sum of each row, column and diagonal [1]. We consider that this characteristic is good balance for unit cell arrange sorting for DAC linearity improvement by cancelling random and systematic mismatch effects among unit cells. We propose unary DAC linearity improvement algorithms by using magic square.

Now we explain about the magic square, which is a square matrix where different integers are arranged to begin with 1 into  $n \times n$ . Furthermore, each row, column, and diagonal components have the same sum. Generally, we call the  $n \times n$  magic square matrix as an  $n$  class magic square. Constant sum of the  $n$  class magic square is expressed as follows:

$$S = \frac{n(n^2 + 1)}{2}$$

We show a  $4 \times 4$  magic square in Fig.3, it can be confirmed that the sum of the elements of each row, column and diagonal components are all equal.

4	9	7	14
16	5	11	2
13	8	10	3
1	12	6	15

Fig.3. Constant sum characteristics of magic square.

## 2.4 Variations in Circuit Element Characteristics

There are systematic variations due to placement (location), and random variations which do not depend on the placement of the circuit element. Ideally, the input and output signals of the DAC should be linear, however due to these variation effects, it becomes nonlinear. The causes of each variation are as follows [2-11].

- 1) Systematic variation
  - Voltage drop on wiring
  - Temperature distribution
  - CMOS manufacturing process
  - a) Doping distribution
  - b) Changes in threshold voltage due to thickness of oxide film
    - Accuracy in wafer plane
    - Mechanical stress
- 2) Random variation
  - Device mismatch

The systematic variation has linear and quadratic gradient variations regarding to the circuit element placement. These are added up and affect circuit operation.

- 1) Linear gradient variation
  - Voltage drop on wiring
  - CMOS manufacturing process
- 2) Quadratic gradient variation
  - Temperature distribution
  - Accuracy in wafer plane
  - Mechanical stress

The above variations largely affect the DAC linearity. The variations  $\varepsilon(x, y)$  are modeled with linear and quadratic variations as well as their combination; they are shown by the following:

- 1) Linear error:  $\varepsilon_l(x, y) = g_l * \cos \theta * x + g_l * \sin \theta * y$   
 $\theta$ : Angle of inclination,  $g_l$ : Magnitude of the slope
- 2) Quadratic error:  $\varepsilon_q(x, y) = g_q * (x^2 + y^2) - a_0$   
 $g_q$ : Variable quantity,  $a_0$ : Position
- 3) Linear and quadratic joint errors:  $\varepsilon_j(x, y) = \varepsilon_l(x, y) + \varepsilon_q(x, y)$

Here,  $(x, y)$  is the coordinates of the position on the chip. Notice that  $(0, 0)$  is not necessarily at the center of the chip; it depends on the modeling of the gradient variation.

## 3. First Technique

### 3.1 Unit Cell Layout Based on Magic Square

The influence of the systematic variation on the linearity of the DAC can be mitigated by the layout technique for the unit cells (Figs. 4, 5). In the case of the unary DAC, the variation effects may be reduced by a random walk method, and then it improves the linearity (Fig. 6) [3, 4].

A magic square has a property that the sums of each row / column / diagonal elements are all equal. Hence we consider that this property balances the unit cell array of the unary type DAC, and we have investigated the layout of the unit cells to reduce the systematic variation effects to improve the DAC linearity.

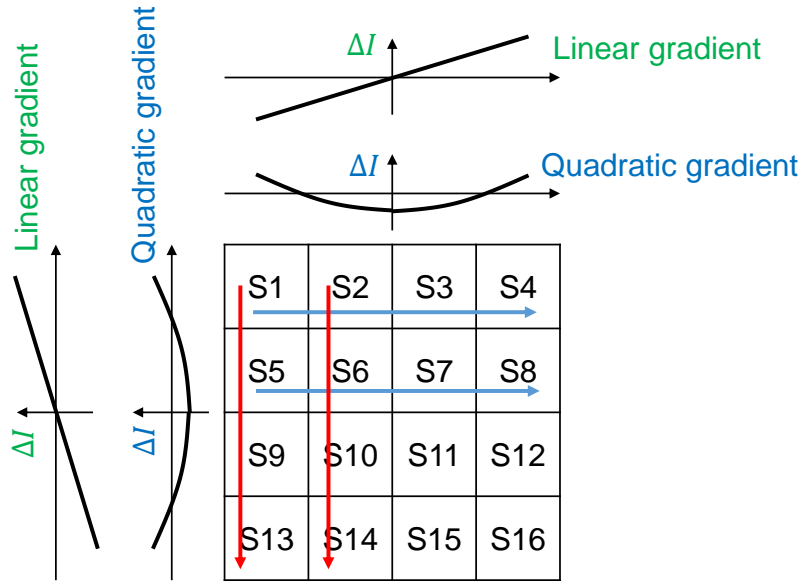
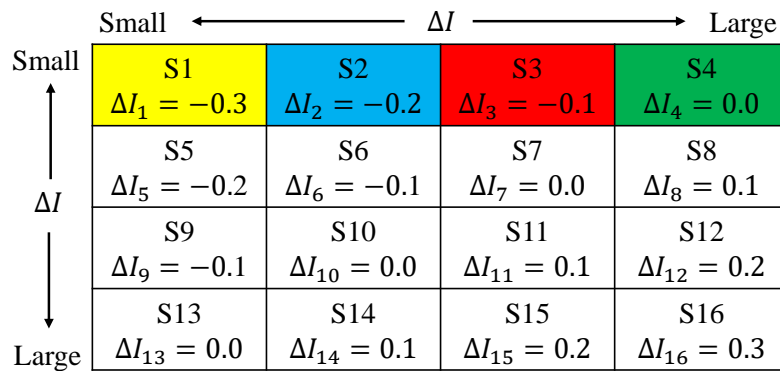
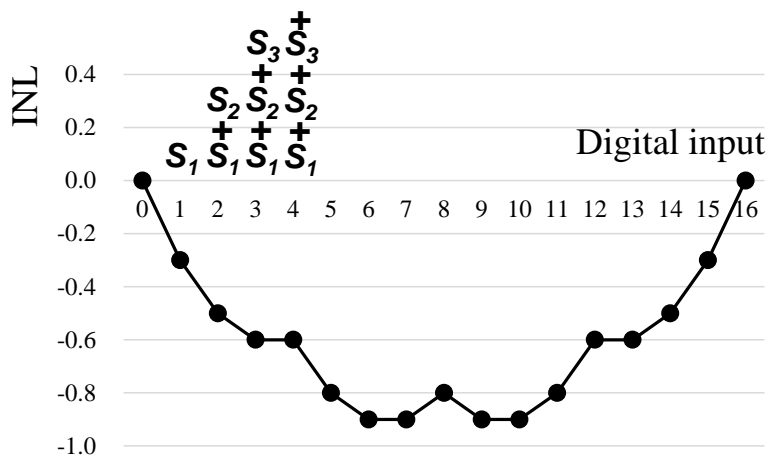


Fig. 4 Regular layout of unit cells for a unary DAC and their linear/quadratic gradient errors



(a)



(b)

Fig. 5. Regular layout (selection method) of unit cells for a unary DAC and its nonlinearity due to their linear gradient errors. (a) Cell selection order and assumed current source mismatches. (b) Simulated INL of the DAC in Fig. 1.



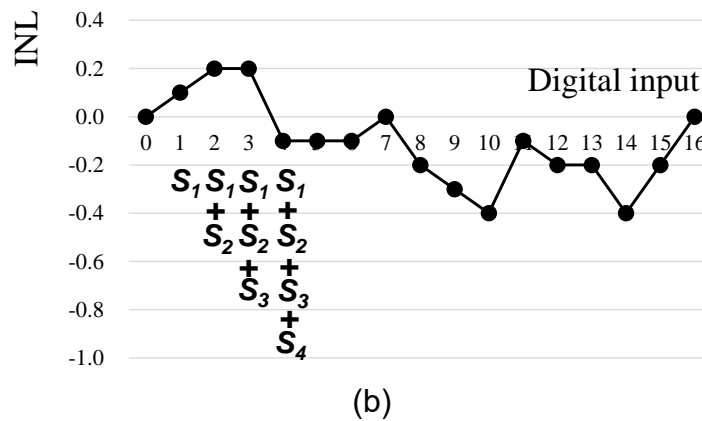
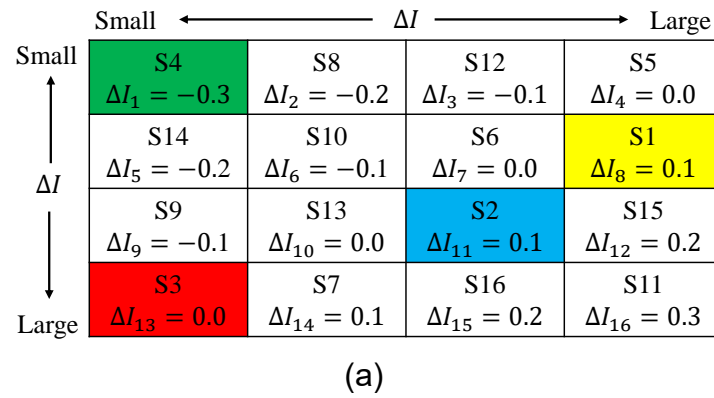


Fig. 6. Random layout (selection method) of unit cells for a unary DAC and its linearity improvement by cancelling their linear gradient errors. (a) Cell selection order and assumed current source mismatches. (b) Simulated INL of the DAC in Fig. 1.

### 3.2 Algorithm Using Concentric Magic Square

Utilizing the constant sum property of the magic square, the systematic variations are expected to be reduced by the magic square layout of the unit cells for a segmented DAC. In the magic square, numbers are placed in a well-balanced manner due to its constant sum property, and hence the unit cell mismatch effects are expected to be cancelled, compared with a random number sorting algorithm.

An 8x8 concentric magic square used in the analysis is shown in Fig. 7. Even if the outside of the magic square by one side is removed in a concentric magic square, the remaining part does not lose its integrity. The combination of four 8x8 concentric magic squares for an 8-bit DAC is used, and the linearity was confirmed by simulation.

### 3.3 Simulation Results and Consideration

#### 1) Linear gradient

The maximum value was set to be  $\pm 1$ , and only the angle  $\theta$  was changed of which the change in INL at that time was examined. In the linear variation, the concentric magic square was effective to reduce the variation effects more than the random walk algorithm.

#### 2) Quadratic gradient

For the quadratic dispersion, the random walk algorithm can reduce nonlinearity better than the concentric magic square one (Fig. 8).

#### 3) Linear and quadratic joint gradient

When the linear gradient variation is larger, the concentric magic square algorithm is suitable (Fig. 9). On the other hand, when the quadratic is larger, the random walk is suitable (Fig. 10).

The magic square has the constant sum properties which have a good balance and a common centroid feature; hence it can take care of the linear error cancellation. We need to select magic squares which can reduce the quadratic error effects further.

59	5	4	62	63	1	8	58
9	18	17	49	50	42	19	56
55	20	28	33	29	40	45	10
54	44	38	31	35	26	21	11
12	43	39	30	34	27	22	53
13	24	25	36	32	37	41	52
51	46	48	16	15	23	47	14
7	60	61	3	2	64	57	6

Fig. 7. 8x8 concentric magic square

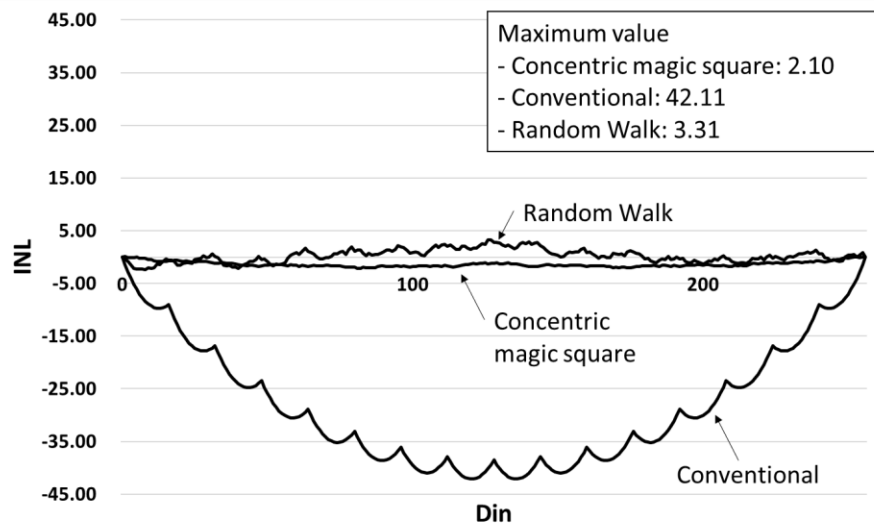


Fig. 8. Simulated DAC digital input (Din) versus INL in case of the quadratic gradient error.

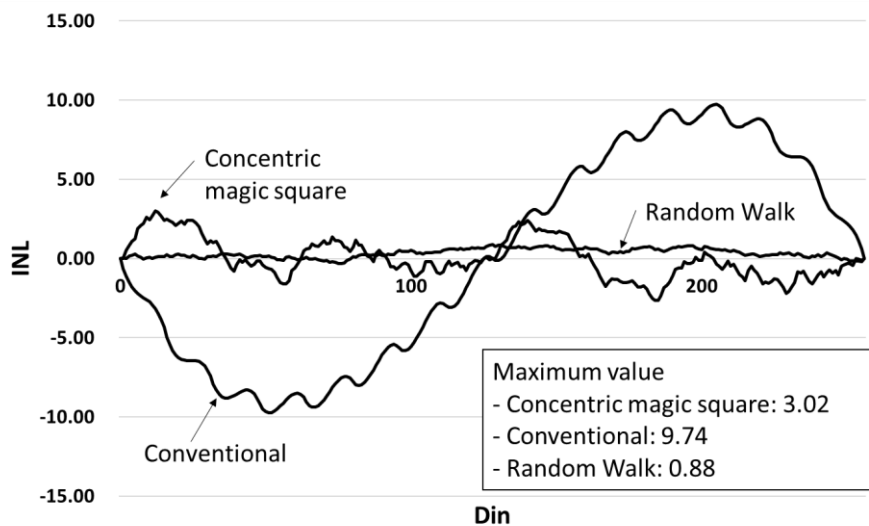


Fig. 9. Simulated Din versus INL when the linear gradient error is bigger than the quadratic gradient error.



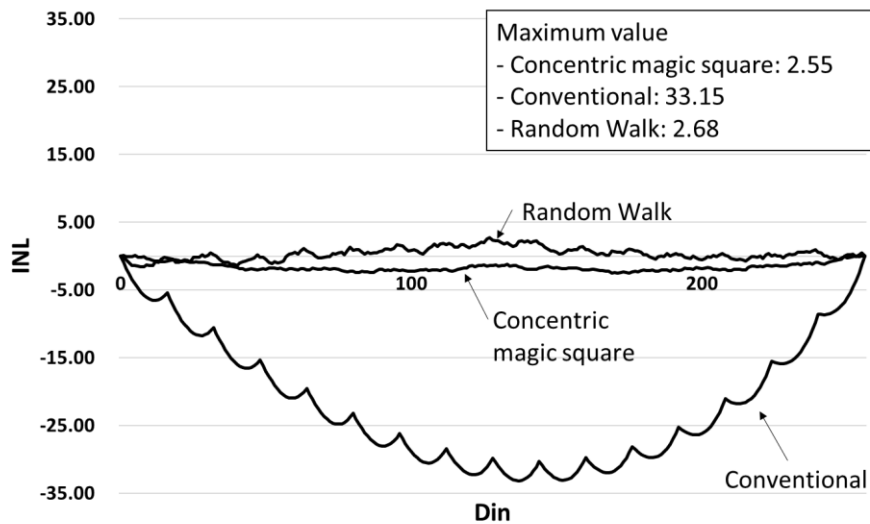


Fig. 10. Simulated Din versus INL when the quadratic gradient error is bigger than the linear gradient error.

## 4. Second Technique

### 4.1 Digital Calibration Algorithm Based on Magic Square

In this section, we propose a digital calibration algorithm based on magic square for the unary current-steering DAC [5-11]. We aim one-time calibration; the new digital algorithm can improve the linearity of the DAC recovering from the effect of the current source mismatches. At the same time, the optimized switching sequence can suppress the distortion especially caused by 2<sup>nd</sup> and 3<sup>rd</sup> order harmonics for better dynamic performance. This section discusses the essence of the proposed digital algorithm as well as the calibration technique (Fig.11).

Note the current measurement circuit only needs to measure the order of the current sources but it does not need their accurate value measurement. Then the measurement circuit can be simply designed as a ring-oscillator-based circuit. [7, 12, 13]

#### 1. Current Source Sorting Based on Magic Square

We use a 4-bit unary DAC example to explain the proposed algorithm.

- 1) Assume the initial condition of current sources after fabrication. (Fig. 11 (a))
- 2) Then, 16 available current source cells are sorted by their values. (Fig. 11 (b))
- 3) The magic square algorithm is adopted for the current source selection, based on the order of the current source values. For example, the smallest current source ( $I_5$ ) is adopted to “1” cell in the magic square. The second small current source ( $I_{15}$ ) is adopted “2” cell in the magic square. Hence, n-th current source ( $I_n$ ) is adopted “n” cell in magic square (Fig.11 (c)).
- 4) A current source cell corresponding to the magic square turns on corresponding to the thermometer-decoded digital input to obtain the DAC analog output (Fig.11 (c)).

We can cancel the random and systematic mismatches by sorting current sources according to constant sum characteristics. We focus on array of the magic square. In the magic square shown in Fig.11 (c), we see at the first column that the sums of adjacent two numbers are almost equal (16 for 1, 15 and 18 for 14, 4)). This is valid also in other columns. Sorting current sources according to constant sum characteristics is implementation of an algorithm to perform successive turn-on switching of current sources with large mismatch and small mismatch, and then we can cancel mismatch effects.

One might consider that its decoder design is an issue, but in Fig.12, we show a look-up table (LUT) of the conventional method as a thermometer-code decoder, and the proposed method as a magic square switching-code decoder. The programmable LUT decoder is relatively easy to implement, thanks to the advancement of LSI technology.

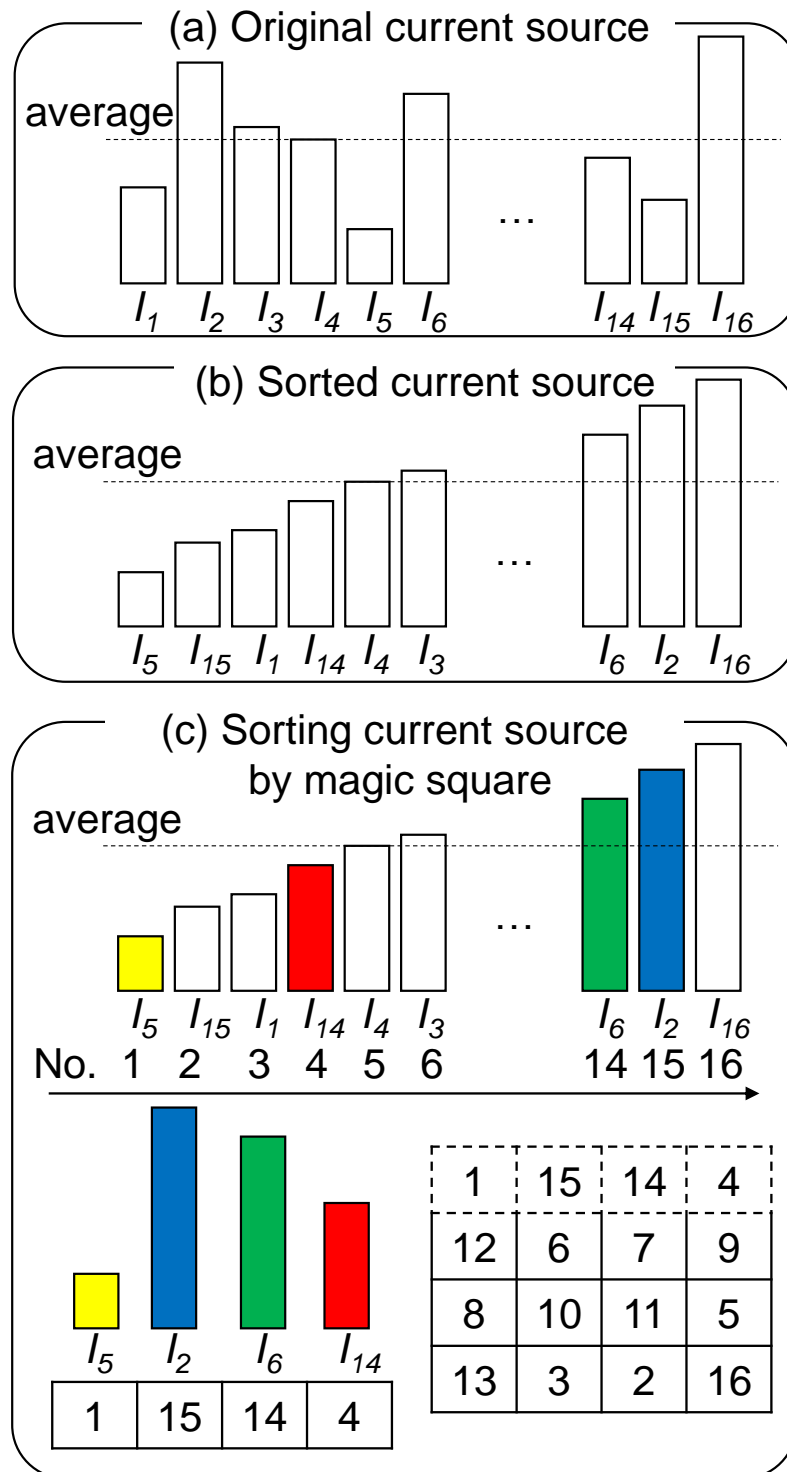
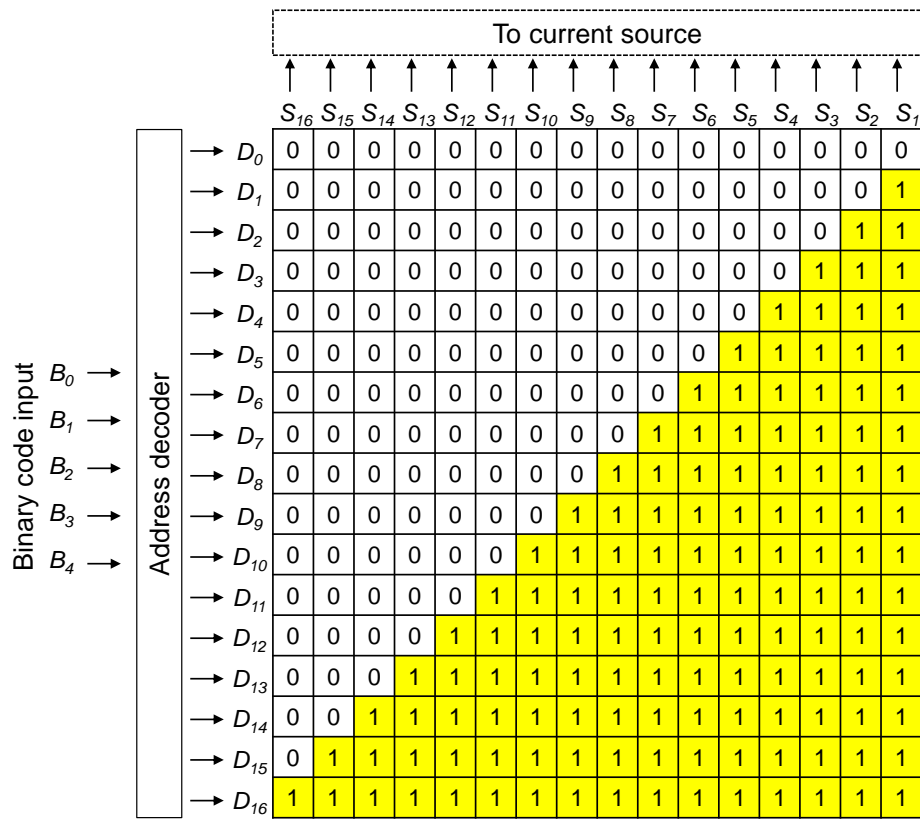
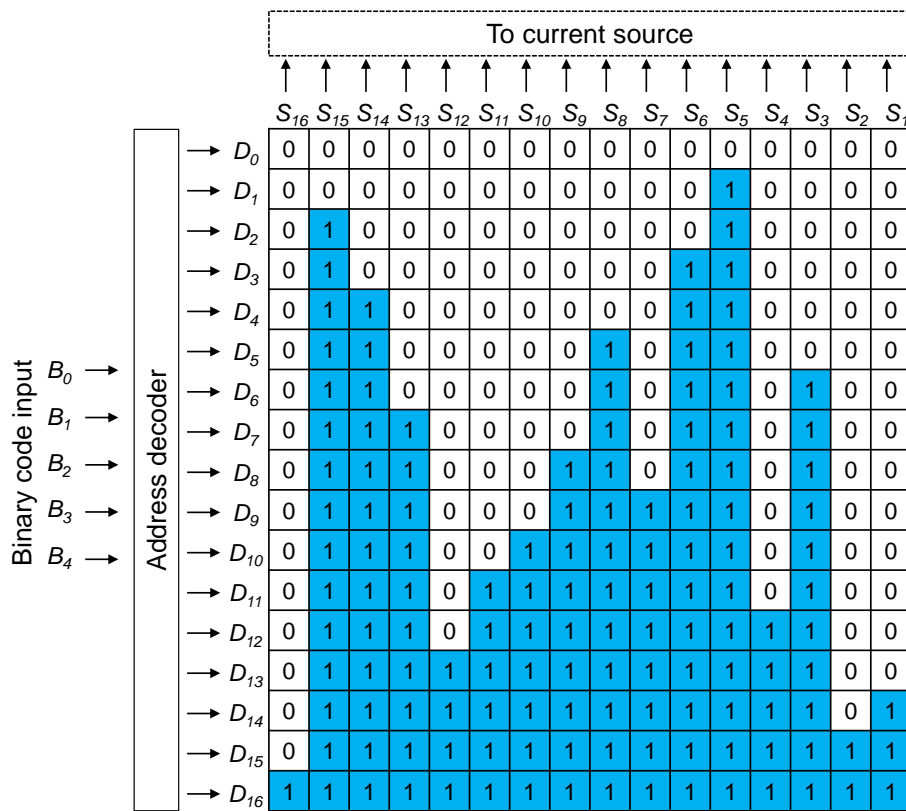


Fig.11. Sorting algorithm based on magic square.



(a) Conventional thermometer-to-binary decoder.



(b) Magic square algorithm decoder.

Fig.12. LUT-based decoder for magic square layout.

## 2. Calibration Technique

We consider here a calibration technique with the error measurement approach to meet with our proposed techniques (Fig.13). Our proposed calibration steps are as follows:

- 1) Input test codes are inserted by central processing unit (CPU) that controls the digital calibration circuit. Digital calibration circuit utilizes ring oscillator-based measurement circuit [3].
- 2) By implementing current measurement circuit, current source cell values are measured.
- 3) All measured values are sorted in memory.
- 4) Then, these data are fed into digital calibration circuit to perform main calibration process.
- 5) Finally, the optimized switching sequence based on magic square is sorted in memory.

The optimized switching sequence is used during DA conversion.

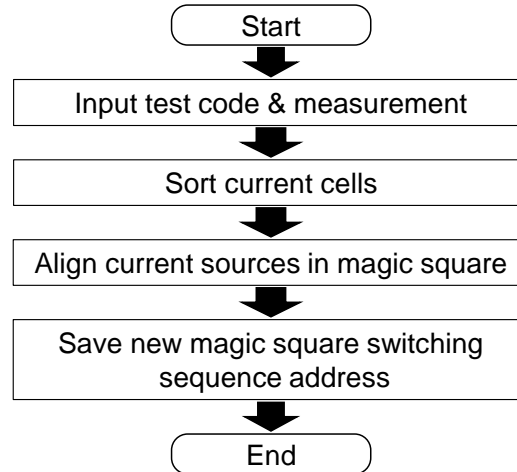


Fig.13. DAC nonlinearity calibration algorithm.

## 4.2 Simulation Results

We have simulated static and dynamic performances of an 8-bit unary DAC. We have compared the conventional method of the thermometer-code decoder and the proposed current source sorting algorithm based on the magic square. The static performance was simulated as INL and DNL. The simulated dynamic performance was obtained as SFDR. In Fig.14, we show the magic square used for the 8-bit unary DAC calibration; this magic square was obtained by MATLAB. Mismatch of current sources was generated as a random number between -1 from +1. We show the simulation results in Figs.15, 16.

### 1. Static Performance

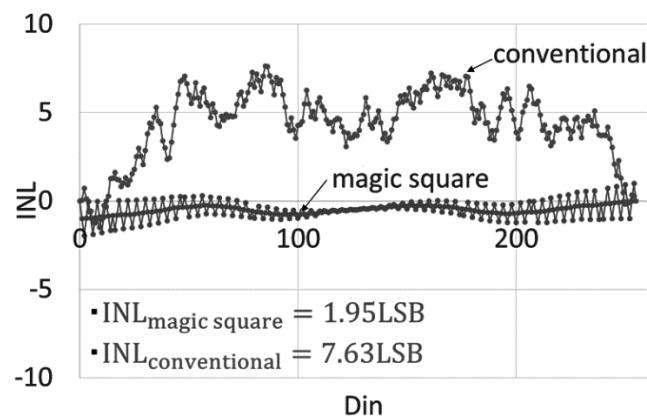
In order to verify the simulated linearity of the proposed DAC, the static performance parameters, INL and DNL have been obtained. The simulation result shows that INL of 7.63 LSB before calibration has been improved to 1.95 LSB using the proposed calibration technique (Fig.15 (a)), and also DNL has been improved (Fig.15 (b)). We note that INL and DNL are nearly 0.0 in the center of the DAC digital input range  $D_{in}$  (=122). This would be due to the fact that many numbers near the middle of the digital input range tend to be placed at the vicinity of the center of the magic square (See Fig. 14 as an example).

### 2. Dynamic Performance

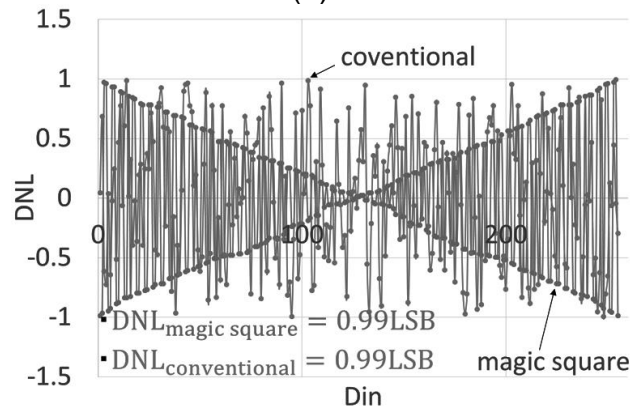
In the frequency domain, the conventional method using the thermometer-code decoder unary 8-bit DAC shows the SFDR performance of 14.8 dB (Fig.16 (a)). Our proposed calibration technique of current source cell sorting based on the magic square obtains SFDR of 22.0dB (Fig.16 (b)), and we see that 7dB improvement of the DAC SFDR using the proposed method is obtained.

256	2	3	253	252	6	7	249	248	10	11	245	244	14	15	241
17	239	238	20	21	235	234	24	25	231	230	28	29	227	226	32
33	223	222	36	37	219	218	40	41	215	214	44	45	211	210	48
208	50	51	205	204	54	55	201	200	58	59	197	196	62	63	193
192	66	67	189	188	70	71	185	184	74	75	181	180	78	79	177
81	175	174	84	85	171	170	88	89	167	166	92	93	163	162	96
97	159	158	100	101	155	154	104	105	151	150	108	109	147	146	112
144	114	115	141	140	118	119	137	136	122	123	133	132	126	127	129
128	130	131	125	124	134	135	121	120	138	139	117	116	142	143	113
145	111	110	148	149	107	106	152	153	103	102	156	157	99	98	160
161	95	94	164	165	91	90	168	169	87	86	172	173	83	82	176
80	178	179	77	76	182	183	73	72	186	187	69	68	190	191	65
64	194	195	61	60	198	199	57	56	202	203	53	52	206	207	49
209	47	46	212	213	43	42	216	217	39	38	220	221	35	34	224
225	31	30	228	229	27	26	232	233	23	22	236	237	19	18	240
16	242	243	13	12	246	247	9	8	250	251	5	4	254	255	1

Fig.14 Magic square used for simulation.

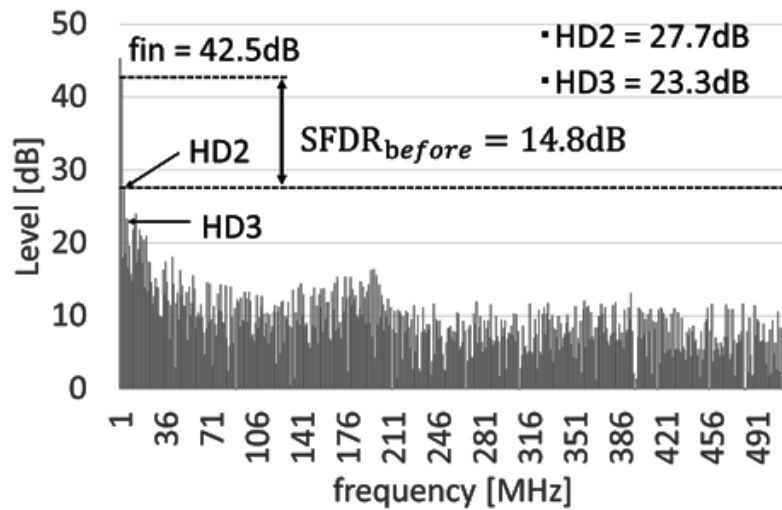


(a) INL

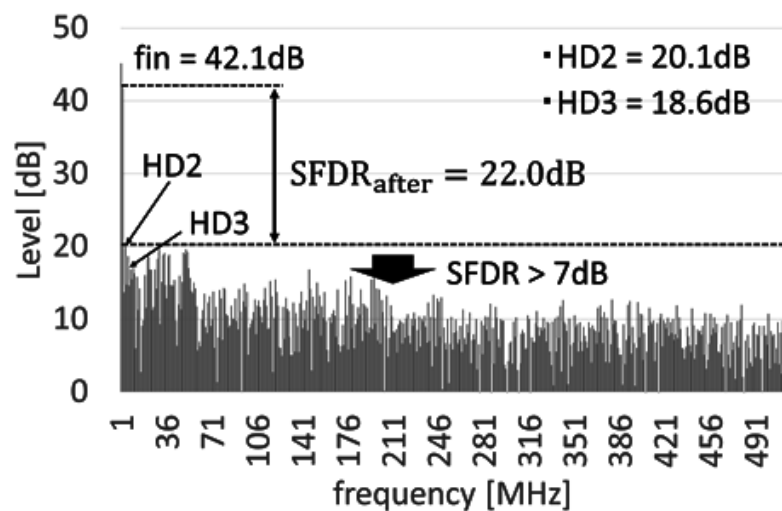


(b) DNL

Fig.15. Simulated DAC static characteristics.



(a) Before calibration.



(b) After the proposed calibration

Fig.16. Simulated DAC output power spectrum.

## 5. Conclusion

We have proposed a layout technique and a current source cell sorting algorithm based on magic square for unary DAC linearity improvement. Our numerical simulation shows that these techniques can improve the unary DAC linearity.

There have been already some similar techniques such as a random walk layout technique to cancel mismatch effects among unit cells, and this paper has shown that there are possibilities to obtain better unary DAC linearity by employing magic square algorithms; this may depend on the mismatch. As the MOSFET size scales down, the mismatch becomes more serious and its characteristic is unknown; this is a big obstacle for high accuracy unary DAC realization. There are a lot of mathematical research results for magic squares and we consider that utilizing these mathematical research fortunes would lead to better linearity improvement algorithms (such as further reduction of the quadrature error effects) for the DAC with nano-CMOS. Also algorithms using Latin squares are expected to be a good switching technique, which has an inherent common-centroid feature [14]. Usage of knight tour algorithm would be another candidate [15].

## Acknowledgements

We would like to thank Dr. Takahiro Miki for valuable discussions.

## References

- [1] K. Omori, The World of Magic Square, *Nippon Hyoron*, 2013.
- [2] M. Pelgrom, C. Duinmaijer, A. Welbers, “Matching properties of MOS transistors”, *IEEE Journal of Solid-State Circuits*, Vol. 24, No.5, pp.1433-1440, 1989.
- [3] T. Miki, Y. Nakamura, M. Nakaya, S. Asai, Y. Akasaka, Y. Horiba, “An 80-MHz 8-bit CMOS D/A Converter”, *IEEE Journal of Solid-State Circuits*, Vol. 21, No. 6, pp.983-988, 1986.
- [4] G. Plas, J. Vandenbussche, W. Sansen, M. Steryaert, G. Gielen, “A14-bit intrinsic accuracy Q2 random walk CMOS DAC,” *IEEE Journal of Solid-State Circuits*, Vol. 34, No. 12, pp. 1708-1718, 1999.
- [5] X. Li, F. Qiao, H. Yang, “Balanced switching schemes for gradient-error compensation in current-steering DACs”, *IEICE Trans. Electron*, Vol. E95-C, No.11, pp. 1790-1798, 2012.
- [6] S. Mohyar, M. Murakami, A. Motozawa, H. Kobayashi, O. Kobayashi, T. Matsuura, “SFDR improvement algorithms for current-steering DACs”, *Key Engineering Materials*, pp. 101-108, 2015.
- [7] S. Mohyar, H. Kobayashi, “Digital calibration algorithm for half-unary current-steering DAC for linearity improvement,” *11th International SoC Design Conference*, (Jeju, Korea), 2014.
- [8] T. Chen, G. Gielen, “A 14-bit 200-MHz current-steering DAC with switching-sequence post-adjustment calibration,” *IEEE J. Solid-State Circuits*, Vol. 42, No. 11, pp.2386-2394, 2007.
- [9] M. Higashino, S. Mohyar, H. Kobayashi, “DAC linearity improvement algorithm with unit cell sorting based on magic square”, *IEEE International Symposium on VLSI Design, Automation and Test*, (Hsinchu, Taiwan), 2016.
- [10] Y. Cong, R. Geiger, “Switching sequence optimization for gradient error compensation in thermometer-decoded DAC arrays”, *IEEE Trans. Circuits and Systems II*, Vol. 47, No. 7, pp.585- 595, 2000.
- [11] K. Kuo, C. Wu, “A switching sequence for gradient error compensation in the DAC design”, *IEEE Trans. Circuits and Systems II*, Vol. 58, No. 8, pp. 502-506, 2011.
- [12] Y. Arakawa, Y. Oosawa, H. Kobayashi, O. Kobayashi, “Linearity improvement technique of multi-bit sigma-delta TDC for timing measurement,” *IEEE 3rd International Workshop on Test and Validation of High-Speed Analog Circuits*, (Anaheim, CA), 2013.
- [13] N. Kushita, J. Kojima, M. Murakami, H. Kobayashi, “Linearity improvement algorithms of multi-bit  $\Delta\Sigma$  DA converter -combination of unit cell re-ordering and DWA”, *2nd International Conference on Technology and Social Science*, (Kiryu, Japan), 2018.
- [14] D. Yao, Y. Sun, M. Higashino, S. N. Mohyar, T. Yanagida, T. Arafune, N. Tsukiji, H. Kobayashi, “DAC linearity improvement with layout technique using magic and Latin squares,” *IEEE International Symposium on Intelligent Signal Processing and Communication Systems*, (Xiamen, China), 2017.
- [15] M. Yenuchenko, A. Korotkov, D. Morozov, “Switching sequence for unary digital-to-analog converters based on knight’s tour”, *IEEE Trans. Circuits and Systems I*, Vol. 66, No. 6, pp.2230-2239, 2019.