ETT-14-53

# DAC Architecture Comparison for SFDR Improvement

**Shaiful Nizam Mohyar***, H. Kobayashi, †

**Gunma University, Japan**
Universiti Malaysia Perlis, Malaysia

# Outline

- Introduction
- Investigated DAC Architecture
- Code Selection Technique
- Simulation Result
- Conclusion

# Outline

- Introduction
- Investigated DAC Architecture
- Code Selection Technique
- Simulation Result
- Conclusion

# Introduction

- **Background**
  - Telecommunication devices
    - Mobile phones, wireless modems & avionics
    - **High-speed, high-accuracy DAC!!!**

- **Transmitter**


Non-idealities !!

DSP → Digital-to-Analog Converter → Filter & Amp → Mixer → Filter & RF Amp → RF Output

Digital — Analog

SFDR degradation!! ☹

Gunma University, Japan

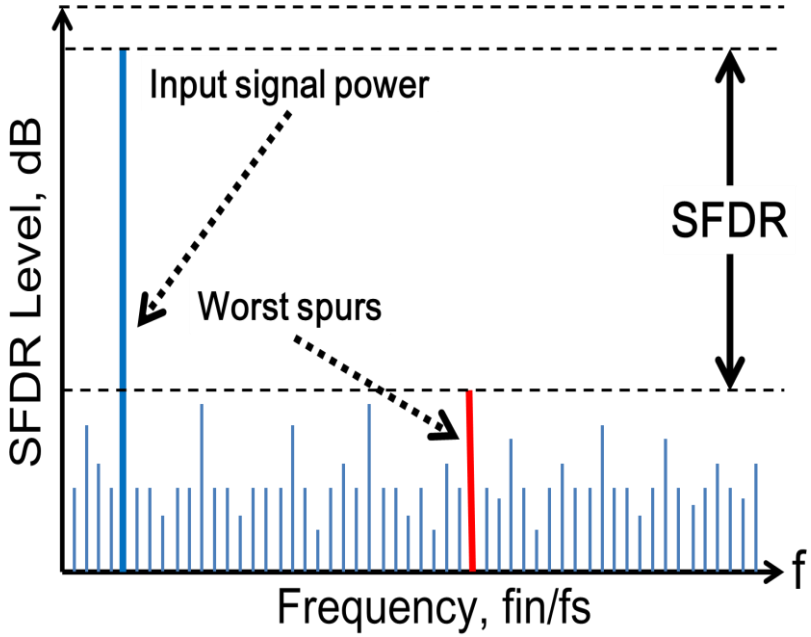
DAC – Digital-to-Analog Converter SFDR – Spurious Free Dynamic Range

# Motivation & Objective

- **Motivation**
  - ➔ Design high SFDR DAC with digital rich configuration
- **Objective**
  - ➔ Reduce interference due to circuit non-idealities
    - ◆ Current source mismatch
- **Approach**
  - ➔ Different DAC architectures & code selection technique

# Spurious Free Dynamic Range (SFDR)

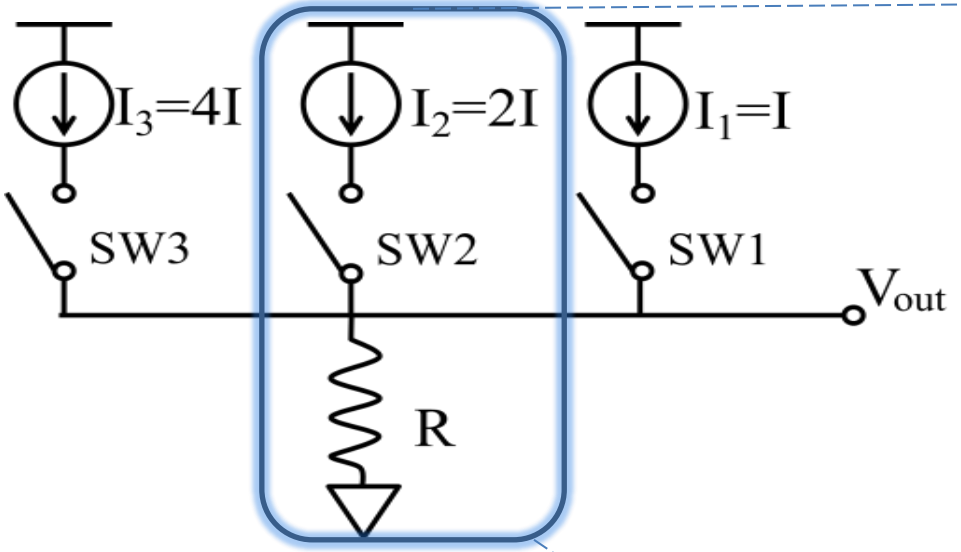
- **Degradation sources**
  - Current source mismatch
    - ➔ Output impedance change
  - Imperfect switch
    - ➔ Temporal disturbance

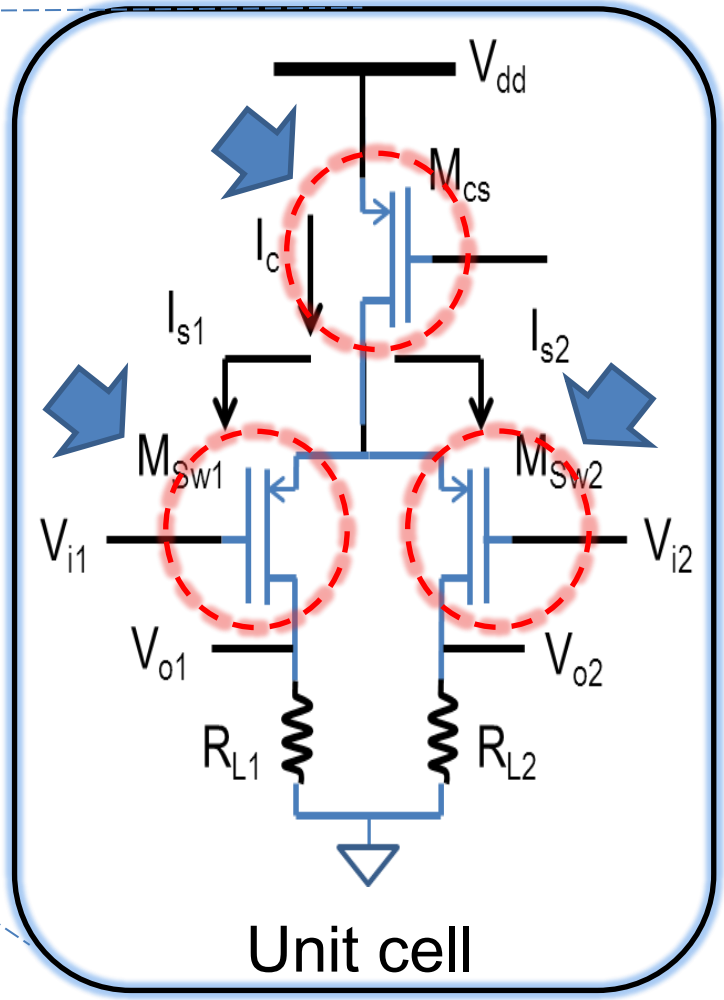


# Investigated Method

- **Method**
  - ➔ Different DAC architecture (Intrinsic redundancy)
    - ➢ **Binary weighted DAC**
    - ➢ **Unary weighted DAC**
    - ➢ **Fibonacci sequence based DAC**
  - ➔ Code selection based on Look-Up Table (LUT))
    - ➢ **Fix (Counter)**
    - ➢ **Random (Randomizer)**
- **Combination**
  - ➔ Dynamic non-linearity improvement

# Current-steering DAC (CS DAC)



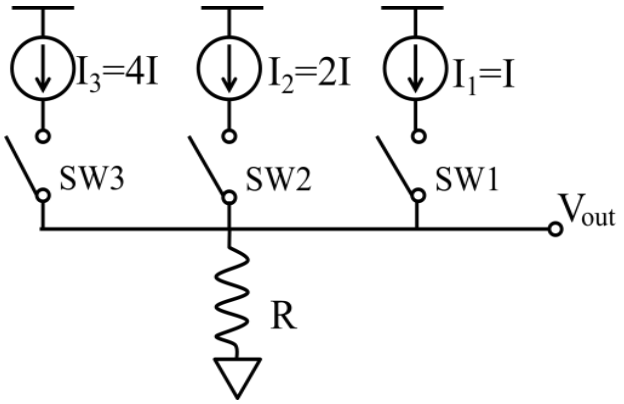

- High-speed
- Easy to integrate
- High-resolution
- Low-power
- Small chip area

# CS DAC limitation

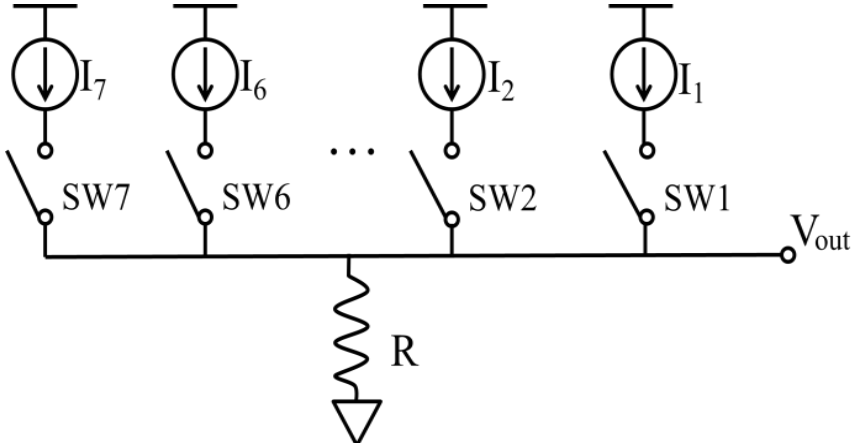
- **Transistor mismatch**
  - Current source mismatch
  - Source of timing errors
- **Mismatch among current cells**
  - Causing DAC static & dynamic non-linearity
- **Better transistor matching**
  - Big size ➔ Power loss
  - Laid out close to each other ➔ Complicated

# Current-steering DAC architecture (1)

- **Binary**
  - Small silicon area ☺
  - High sampling speed ☺
  - Large glitch energy ☹
  - No redundancy ☹

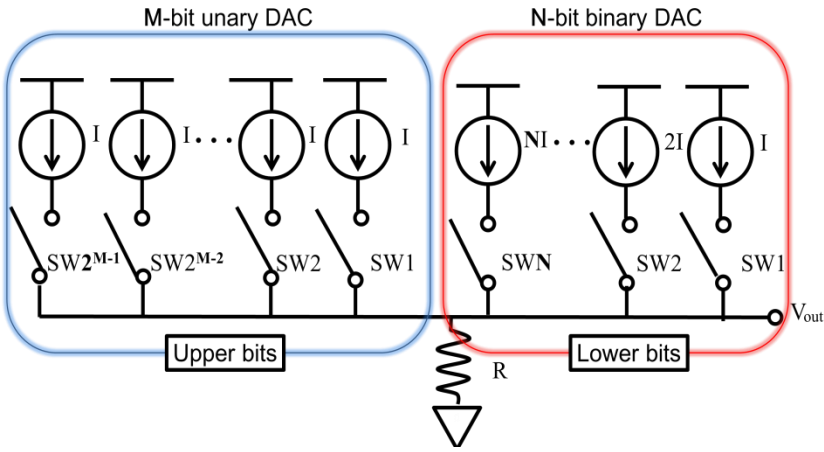



- **Unary / Thermometer-coded (TC)**
  - Small glitch energy ☺
  - High sampling speed ☺
  - Redundancy ☺
  - Large silicon area ☹

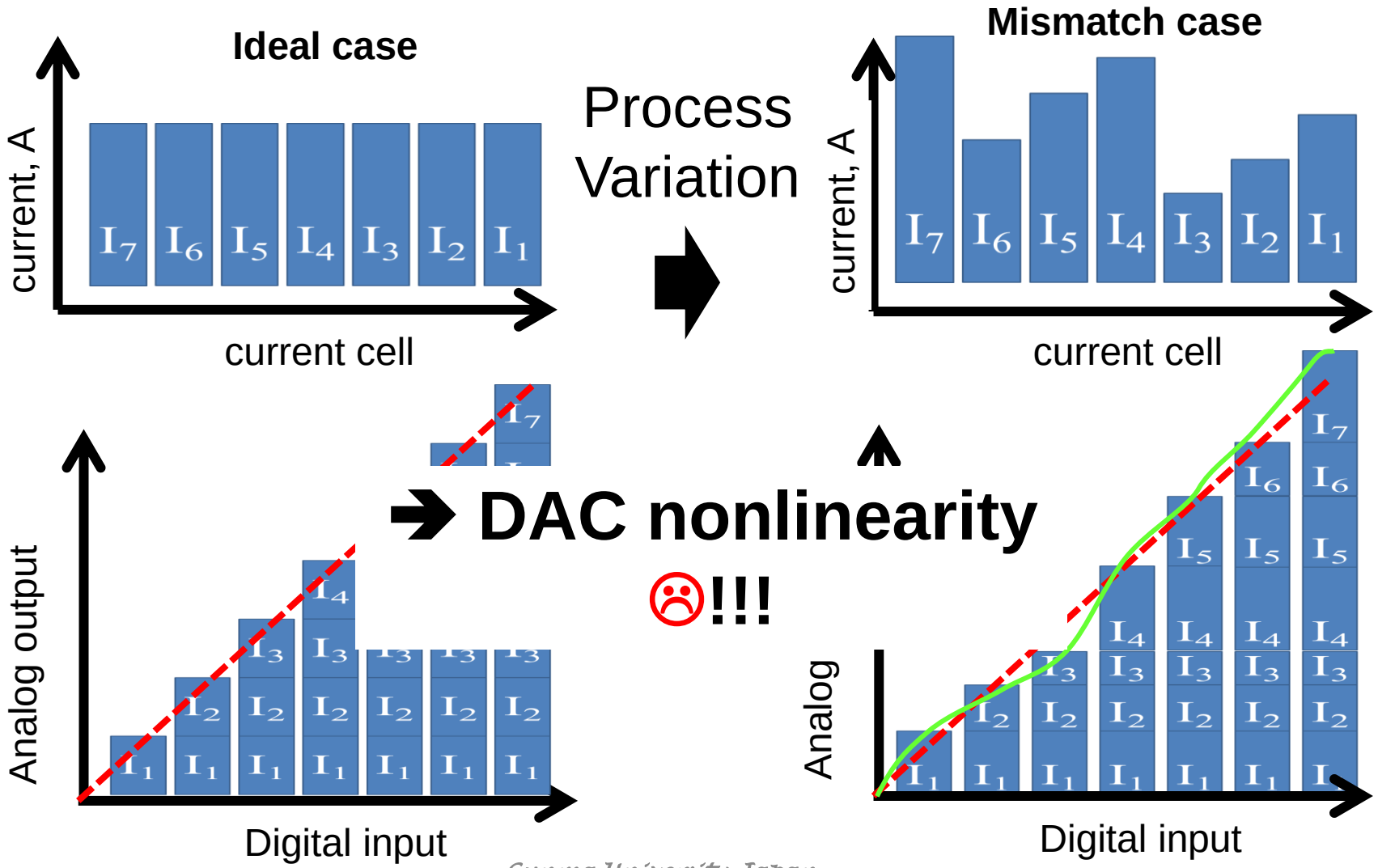


# Current-steering DAC architecture (2)

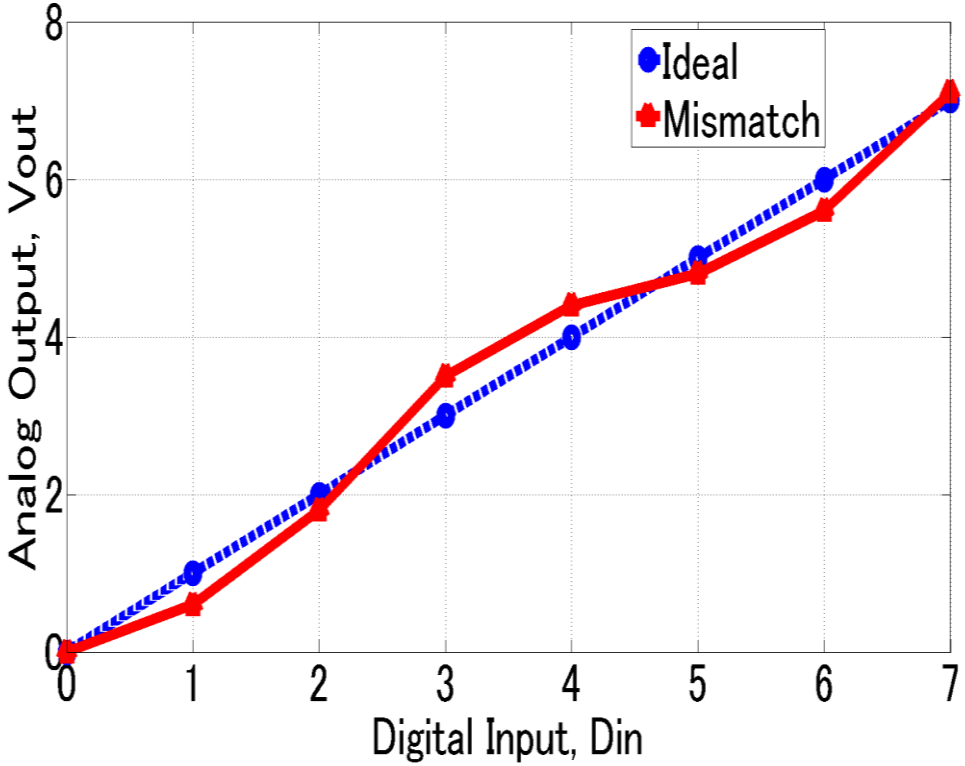
- **Segmented**
  - Balanced performance ☺
  - Complex ☹

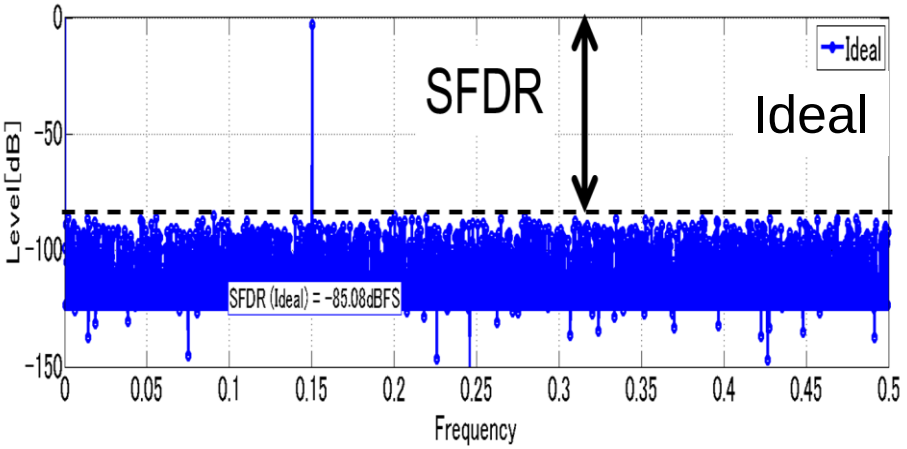


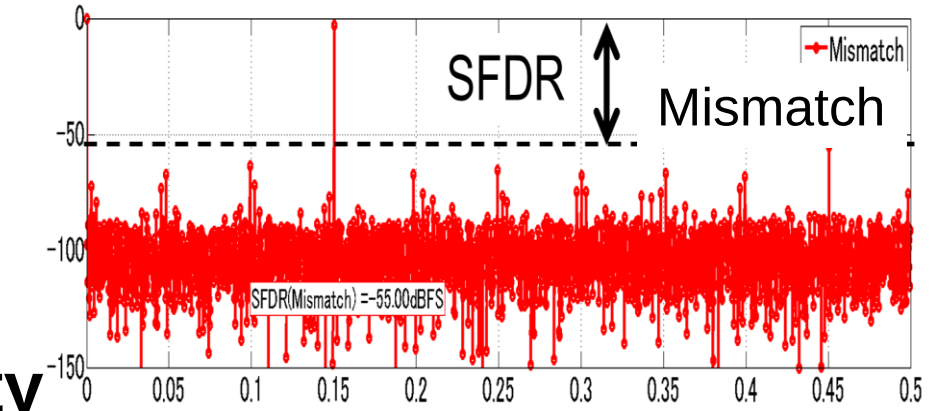
# Current Source Mismatch



# Nonlinearity & SFDR degradation




**CS Mismatch ➔ Nonlinearity**
**SFDR degradation**

CS – Current Source

SFDR – Spurious Free Dynamic Range

# Outline

- Introduction
- Investigated DAC Architecture
- Code Selection Technique
- Simulation Result
- Conclusion

# Fibonacci sequence

- **Fibonacci sequence**

$$F_0 = 0; F_1 = 1;$$
$$F_2 = F_0 + F_1$$
$$F_{n+2} = F_n + F_{n+1} \quad n \geq 0$$

- **Generally**

$$F_n = 0, 1, 1, 2, 3, 5, 8, 13, \ldots$$

- **Ratio**

$$\lim_{n \to \infty} \frac{F_n}{F_{n-1}} \approx 1.6$$

| Unary < Fibonacci < Binary | | |
|---|---|---|
| **1** | **1.6** | **2** |

# Redundancy

- Step ➔ occupied silicon area
- Code combination ➔ output optimization

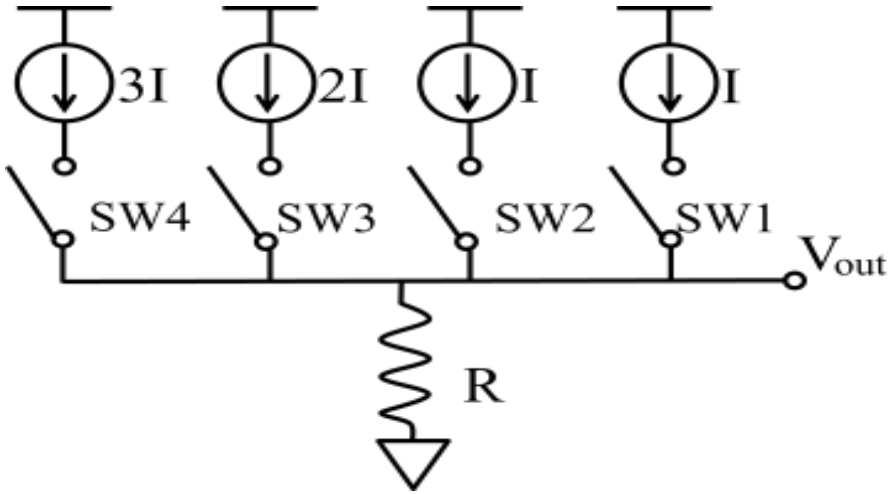
| Comparison between Binary, Unary & Fibonacci code | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Binary ($B_n$) | | | Unary ($U_n$) | | | Fibonacci ($F_n$) | | |
| Bit, n | Weights, $w_n$ | Step, n | Combination $2^n$ (max) | Weights, $w_n$ | Step, $2^n-1$ | Combination $2^{2n-1}$ (max) | Weights, $w_n$ | Step, k | Combination (max) |
| 2 | 2,1 | 2 | 4 (1) | $1_3,1_2,1_1$ | 3 | 8 (3) | 2,1,1 | 3 | 7 (2) |
| 3 | 4,2,1 | 3 | 8 (1) | $1_7,\ldots,1_1$ | 7 | 128 (35) | 3,2,1,1 | 4 | 16 (3) |
| 4 | 8,4,2, 1 | 4 | 16 (1) | $1_{15},\ldots,1_1$ | 15 | 32768 (6435) | 8,5,3, 2,1,1 | 6 | 51 (5) |
| 5 | 16,8,4, 2,1 | 5 | 32 (1) | $1_{31},\ldots,1_1$ | 31 | $2.1\text{x}10^9$ ($3.0\text{x}10^8$) | 13,8,5, 3,2,1,1 | 7 | 126 (6) |

**Binary < Fibonacci < Unary**

(Area) ( Area & Redundancy) (Redundancy)

# Investigated DAC

- **Fibonacci sequence based DAC**

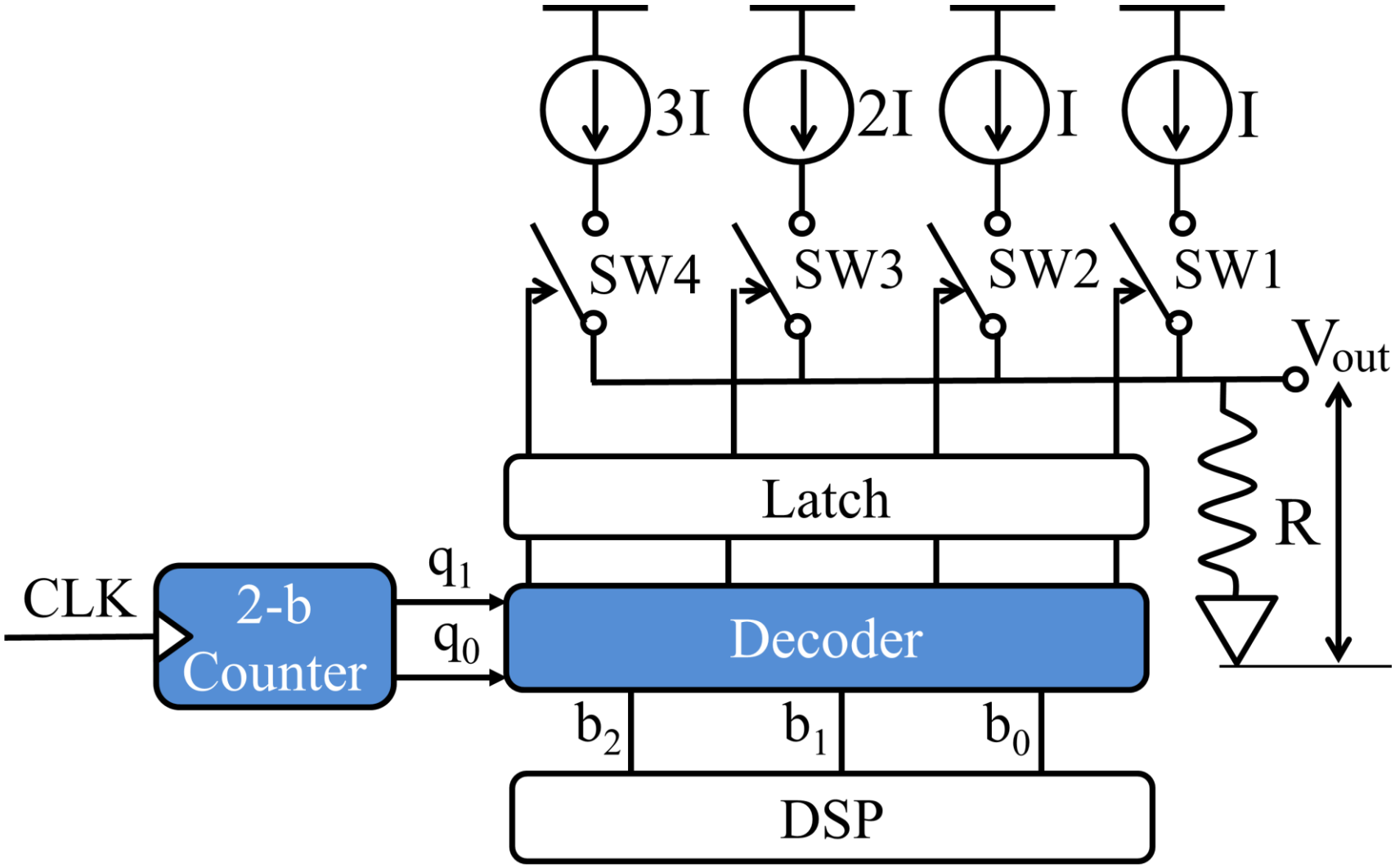



- Small silicon area ☺
- Redundancy ☺
- Large glitch energy ☹

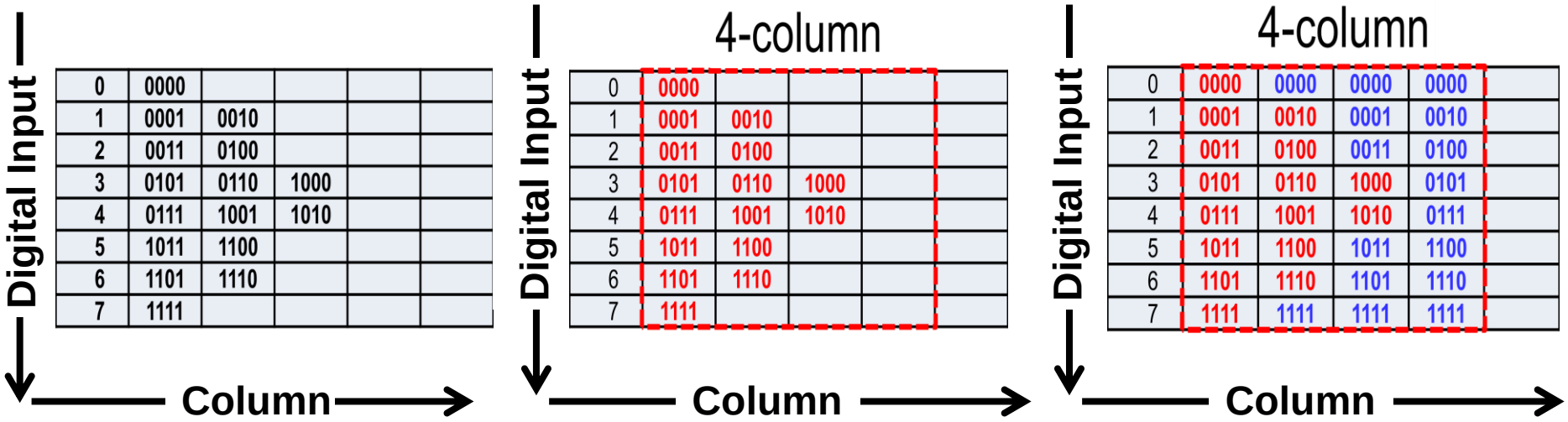
# Outline

- Introduction
- Investigated DAC Architecture
- Code Selection Technique
- Simulation Result
- Conclusion

# 1. Fixed Counter (Fibonacci)



# Decoder (LUT)




| | | | | | |
|---|---|---|---|---|---|
| 0 | 0000 | | | | |
| 1 | 0001 | 0010 | | | |
| 2 | 0011 | 0100 | | | |
| 3 | 0101 | 0110 | 1000 | | |
| 4 | 0111 | 1001 | 1010 | | |
| 5 | 1011 | 1100 | | | |
| 6 | 1101 | 1110 | | | |
| 7 | 1111 | | | | |

| | | | | |
|---|---|---|---|---|
| 0 | 0000 | | | |
| 1 | 0001 | 0010 | | |
| 2 | 0011 | 0100 | | |
| 3 | 0101 | 0110 | 1000 | |
| 4 | 0111 | 1001 | 1010 | |
| 5 | 1011 | 1100 | | |
| 6 | 1101 | 1110 | | |
| 7 | 1111 | | | |

| | | | | |
|---|---|---|---|---|
| 0 | 0000 | 0000 | 0000 | 0000 |
| 1 | 0001 | 0010 | 0001 | 0010 |
| 2 | 0011 | 0100 | 0011 | 0100 |
| 3 | 0101 | 0110 | 1000 | 0101 |
| 4 | 0111 | 1001 | 1010 | 0111 |
| 5 | 1011 | 1100 | 1011 | 1100 |
| 6 | 1101 | 1110 | 1101 | 1110 |
| 7 | 1111 | 1111 | 1111 | 1111 |

① Generate a possible code combinations

② Decide number of column

③ Add empty column by repeating existed code

# Operational (Fix)

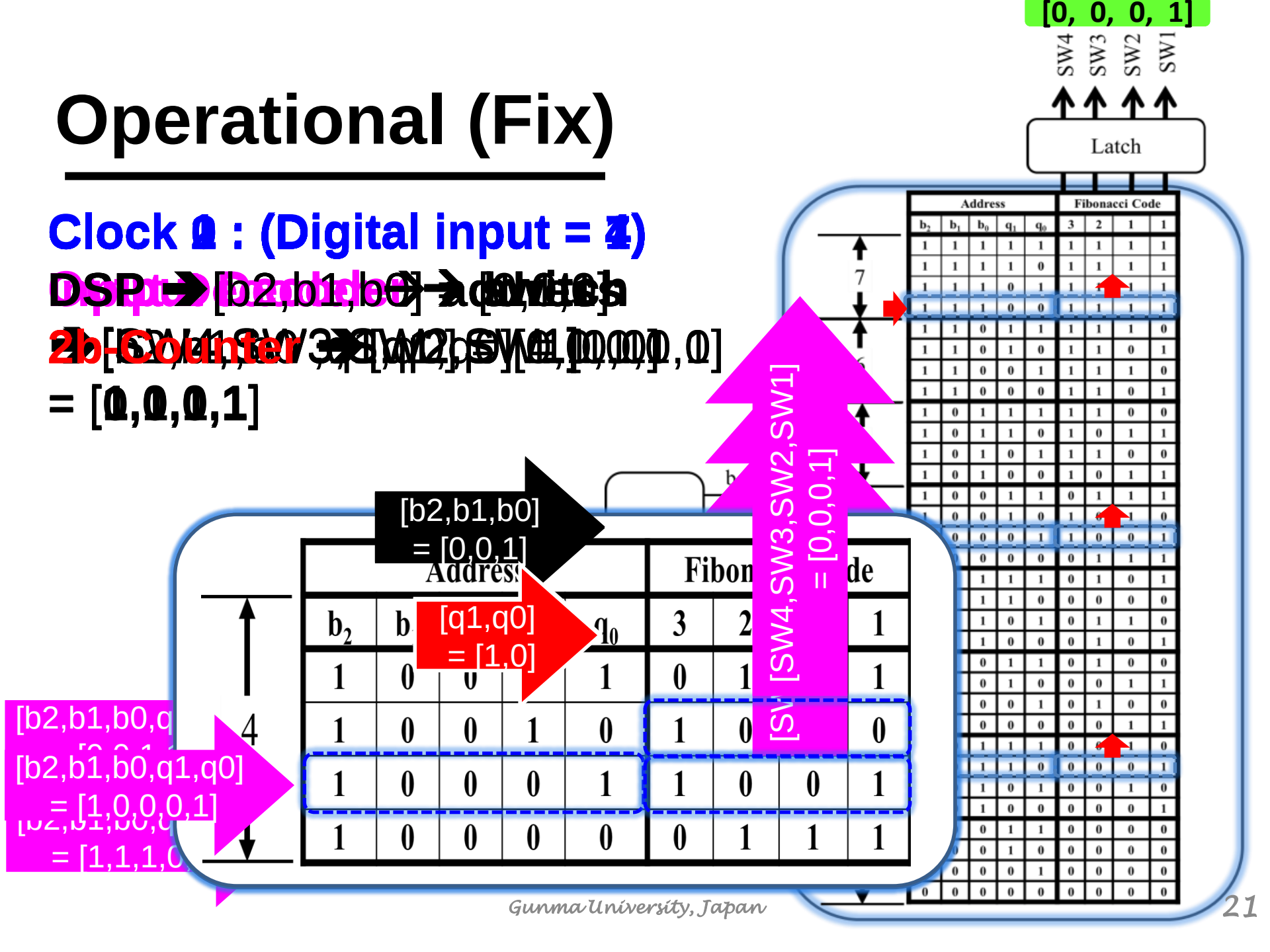
**Clock 1 : (Digital input = 4)**

DSP → [b2,b1,b0] → Address

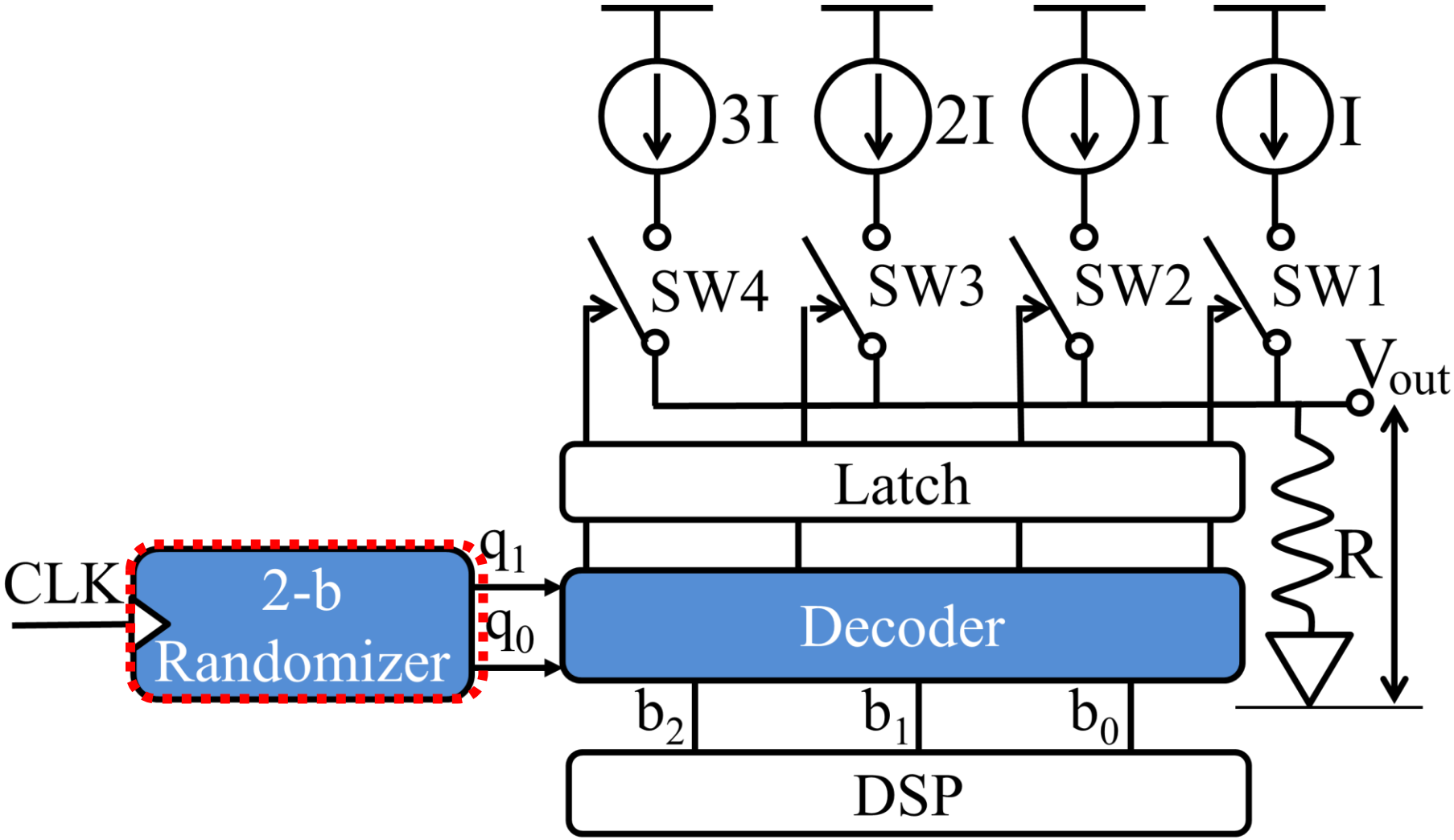
Input → Decoder

2b-Counter → [q1,q0]

= [1,0,0,0,1]



# 2. Randomizer (Fibonacci)



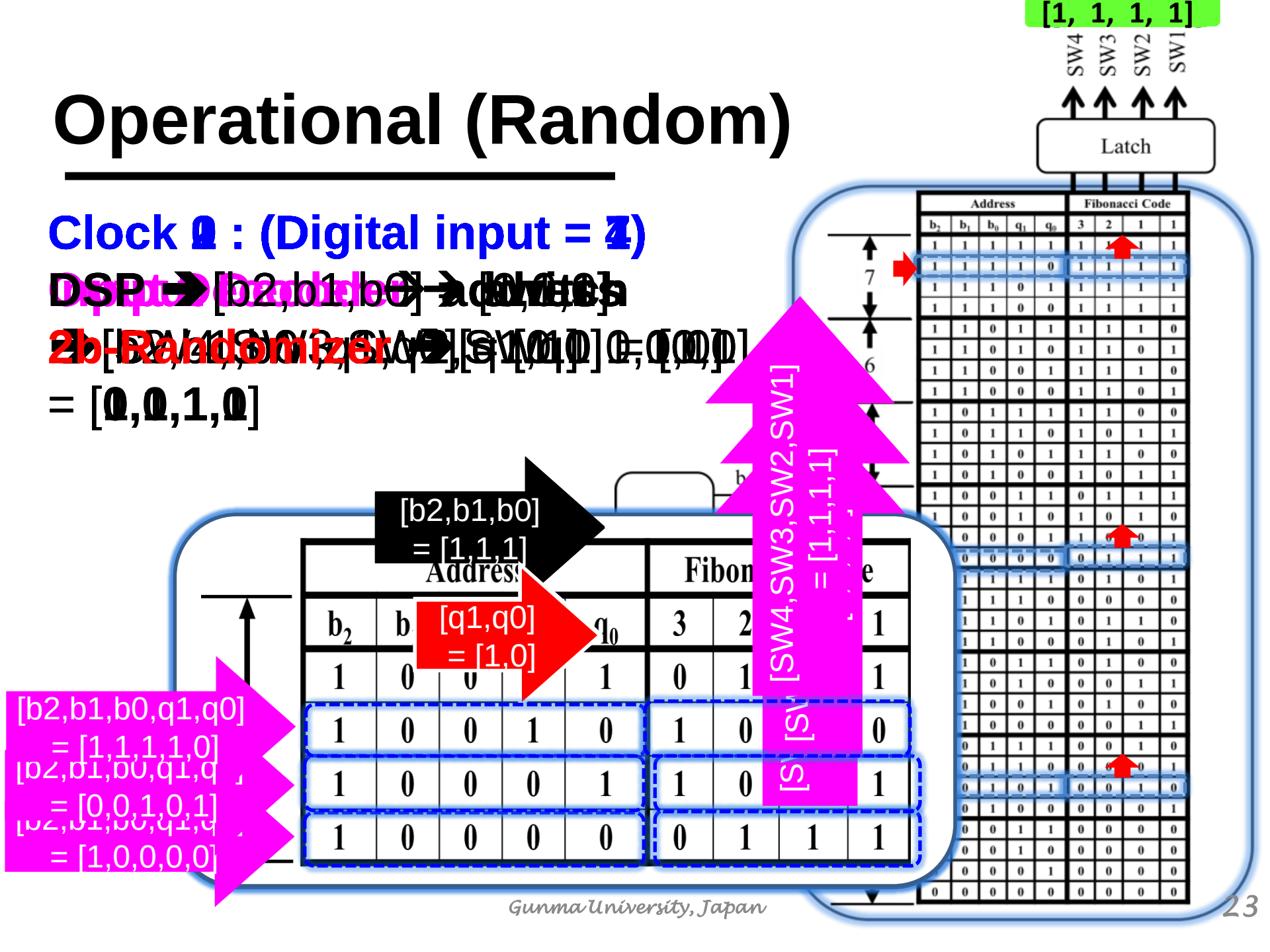
# Operational (Random)

**Clock 1 : (Digital input = 4)**

DSP → [b2,b1,b0] → address

2b-Randomizer → [q1,q0] → address

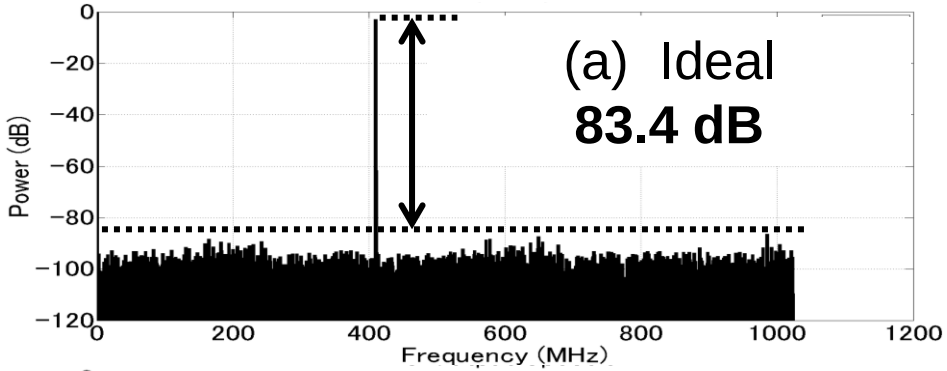
= [0,0,1,0]



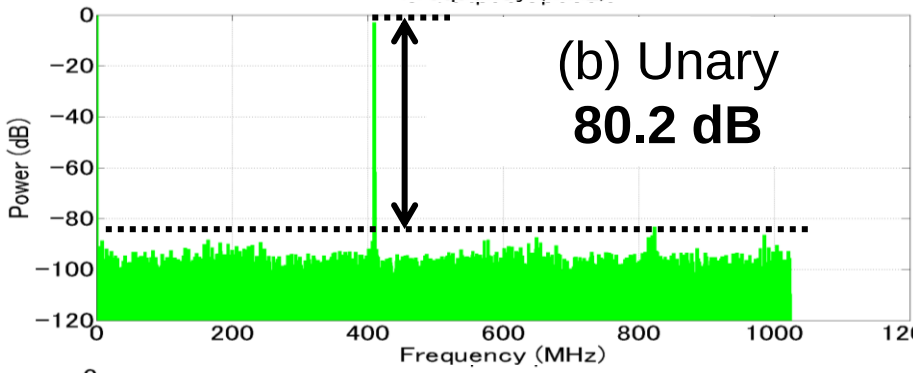
# Outline

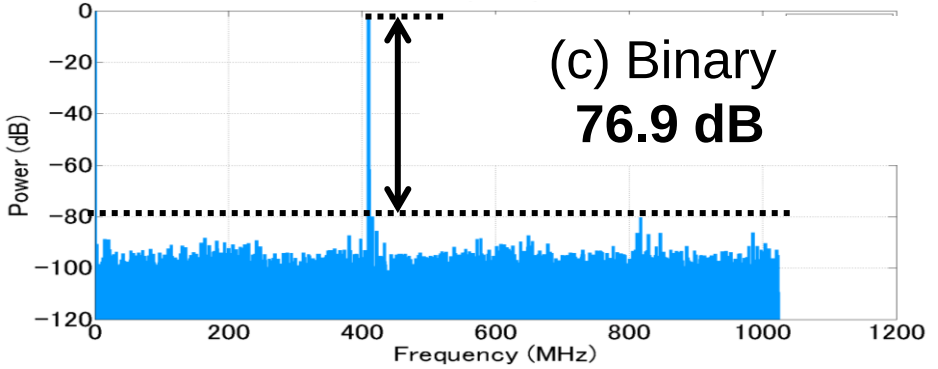
- Introduction
- Investigated DAC Architecture
- Code Selection Technique
- Simulation Result
- Conclusion

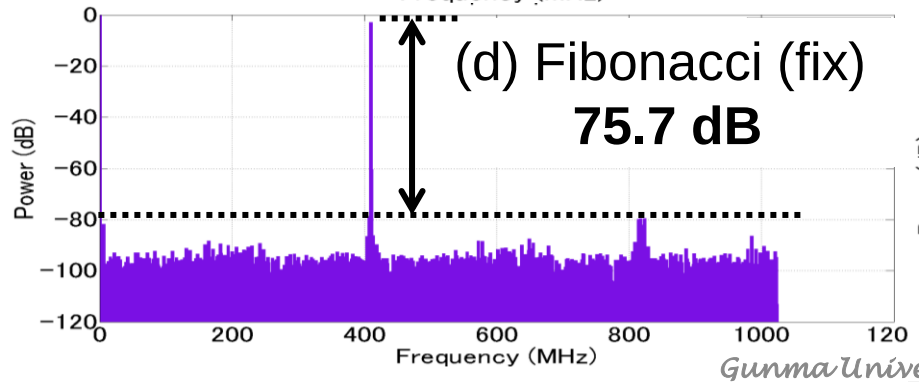
# SFDR Performance

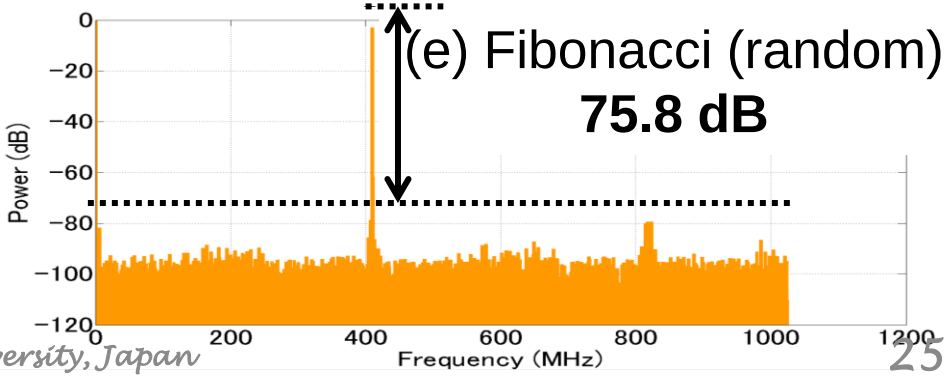



10-bit
fin = 400MHz, fs = 2GHz,
mismatch: 10% ,
min:-0.05, max:0.05

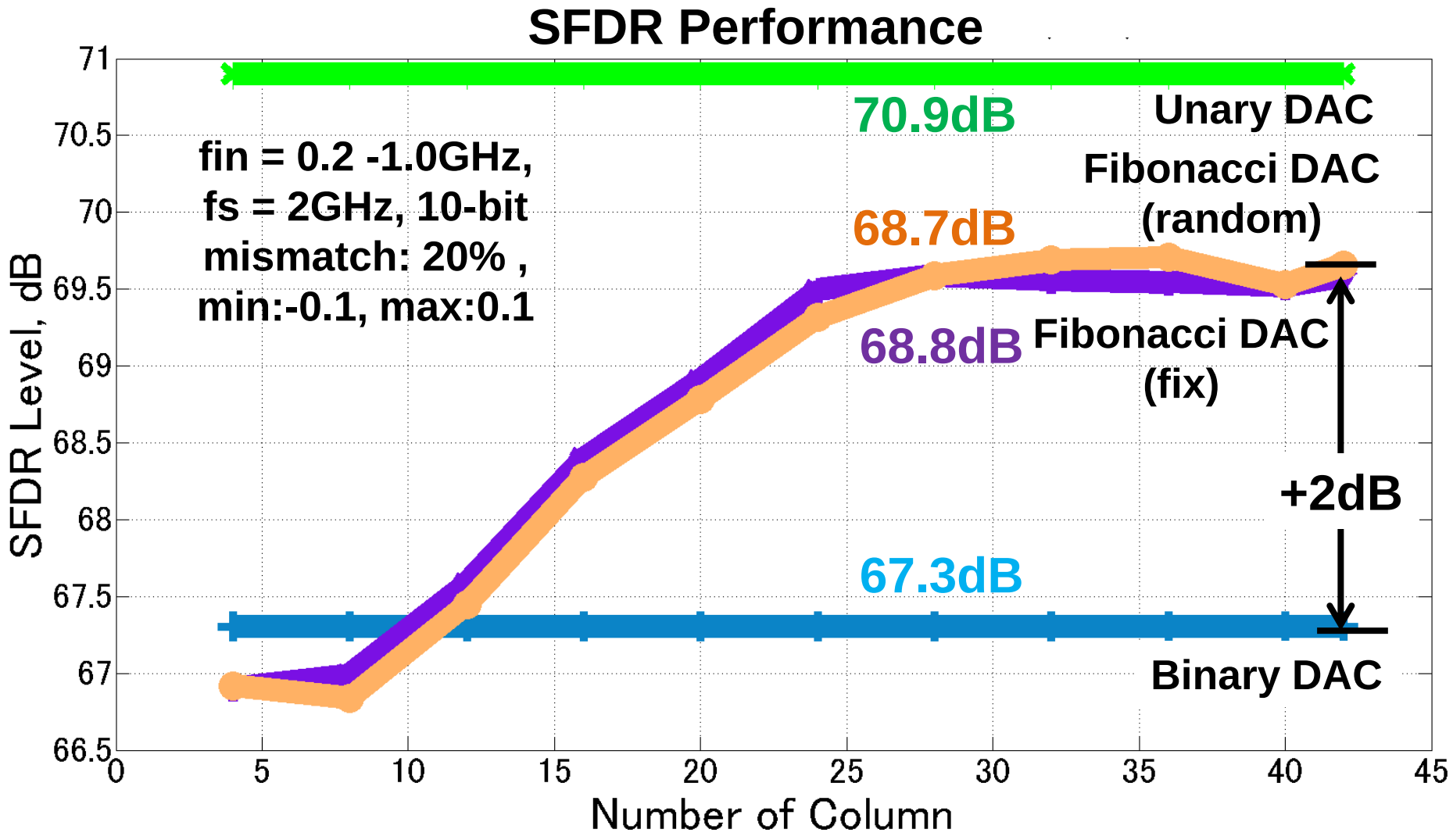


# Number of Column



# Outline

- Introduction
- Investigated DAC Architecture
- Code Selection Technique
- Simulation Result
- Conclusion

# Conclusion

- SFDR performance (4 columns selection)
  - Unary > Binary > **Fibonacci**
  - Fibonacci (Fix & Random) ≈ Binary

- SFDR performance ( > 4 columns selection)
  - Fibonacci (Fix & Random) > Binary ≈ **+2dB**
  - Unary > **Fibonacci** > Binary

- Future work
  - Need more proper code arrangement

**Thank you**
**very much**
**for your kindly**
**attention**

# Q & A

Q: To compare the redundancy, why not compared with other existed architectures which also use redundancy rather than compare with conventional binary architecture?