# Time-to-Digital Converter Architecture with Residue Arithmetic and its FPGA Implementation

Congbing Li Kentaroh Katoh Junshan Wang Shu Wu Shaiful Nizam Mohyar Haruo Kobayashi
Division of Electronics and Informatics, Gunma University 1-5-1 Tenjin-cho Kiryu Gunma 376-8515 Japan
Phone: 81-277-30-1789 fax: 81-277-30-1707 e-mail: t13802483@gunma-u.ac.jp
Tsuruoka National College of Technology, Japan email: k-katoh@tsuruoka-nct.ac.jp

**Abstract**

This paper describes a time-to-digital converter (TDC) architecuture with residue arithmetic or Chinese Remainder theorem. It can reduce the hardware and power significantly compared to a flash type TDC while keeping comparable performance. Its FPGA implementation and measurement resuts show the effectiveness of our proposed architecture.

***Keywords- Timing Measurement, Time to Digital Converter, Residue, Chinese Remainder Theorem, FPGA***

## Introduction

A Time-to-Digital-Converter (TDC) measures the time interval between two edges, and time resolution of several picoseconds can be achieved when the TDC is implemented with an advanced CMOS process. TDC applications include phase comparators of all-digital PLLs, sensor interface circuits, modulation circuits, demodulation circuits, as well as TDC-based ADCs. The TDC will play an increasingly important role in the nano-CMOS era, because it is well suited to implementation with fine digital CMOS processes. [1,2,3].

There are various kinds of TDC circuits, and here we focus on a flash-typeTDC (Fig.1) [1]. It uses a delay line which consists of CMOS inverter buffer delays. Baesd on this flash-type TDC, we will introduce a new type TDC---Residue Arithmetic TDC to reduce the hardware and power significantly compared to a flash-type TDC while keeping comparable performance.Then we have implemtened it on an FPGA to verify the operation and performance.

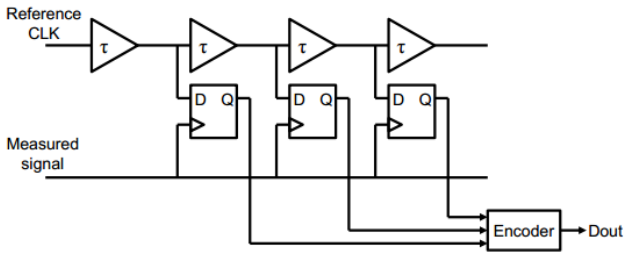



Fig 1. Flash-type TDC

## Residue Arithmetic

Suppose that m1,…,mr are positive integers and coprime each other. Then there is unique positive interger x for given integers (a1,…,ar) which satisfy the following:

$$x \equiv a_k \pmod{m_k}, \quad k = 1, 2, \ldots, r$$

where $0 \leqq a_k < m_k$, $0 \leqq x < N$ ($N = m_1 \cdot m_2 \cdots m_r$). Table I shows the case of m1= 2, m2= 3, m3 = 5 and N=2 x 3 x 5=30, and we see that each k is mapped to residues of (m1, m2, m3 ) one to one [3,4].

Table I. An integer k and residues of (m1, m2, m3 )

| m1 | m2 | m3 | k | m1 | m2 | m3 | k |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 15 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 16 |
| 0 | 2 | 2 | 2 | 1 | 2 | 2 | 17 |
| 1 | 0 | 3 | 3 | 0 | 0 | 3 | 18 |
| 0 | 1 | 4 | 4 | 1 | 1 | 4 | 19 |
| 1 | 2 | 0 | 5 | 0 | 2 | 0 | 20 |
| 0 | 0 | 1 | 6 | 1 | 0 | 1 | 21 |
| 1 | 1 | 2 | 7 | 0 | 1 | 2 | 22 |
| 0 | 2 | 3 | 8 | 1 | 2 | 3 | 23 |
| 1 | 0 | 4 | 9 | 0 | 0 | 4 | 24 |
| 0 | 1 | 0 | 10 | 1 | 1 | 0 | 25 |
| 1 | 2 | 1 | 11 | 0 | 2 | 1 | 26 |
| 0 | 0 | 2 | 12 | 1 | 0 | 2 | 27 |
| 1 | 1 | 3 | 13 | 0 | 1 | 3 | 28 |
| 0 | 2 | 4 | 14 | 1 | 2 | 4 | 29 |

## Residue Arithmetic TDC Architecture

We consider to use this residue arithmetic for TDC implementation, because obtaining the residue is relatively easy for time signal (used in TDC design) while it is difficult for voltage signal. (used in ADC design). Fig.2 shows the proposed residue arithmetic TDC in the case of m1= 2, m2= 3, m3 = 5 and N=2 x 3 x 5=30, where the residues a (mod 2), b

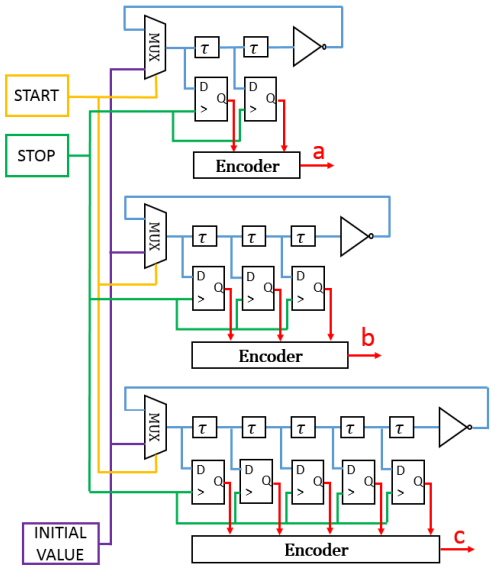



Fig2. Proposed residue arithmetic TDC architecture.

(mod 3), c (mod 5) are obtained with ring oscillators.

Note that the proposed TDC uses only 10 delay cells and 10 flip-flops (because 2+3+5=10) while the corresponding flash TDC requires 30 delay cells and 30 flip-flops; in general, the proposed TDC uses M delay cells and M flip-flops (where M=m1+m2+···+mr) while the corresponding flash-type TDC uses N delay cells and N flip-flops (where N = m1·m2···mr), and hence the circuit and power reduction of the proposed TDC can be significant for a large N with proper choice for M<<N compared to the flash TDC.

## FPGA Implementation

We have implemented our proposed TDC with an FPGA (Fig.3) [5, 6, 7], and Table II and Fig.4 show its measurement results. We see that the proposed TDC works with good linearity as expected.

## Conclusion

This paper describes residue arithmetic TDC and its FPGA implementation, and the measurement results verify its operation principle.


## Acknowledgements

We would like to thank STARC which supports this project.


## References


[1] S. Ito, S. Nishimura, H. Kobayashi, S. Uemori, Y. Tan, N. Takai, T. Yamaguchi, K. Niitsu,"Stochastic TDC Architecture with Self-Calibration," IEEE Asia Pacific Conference on Circuits and Systems (Dec. 2010).

[2] K. Katoh, Y. Doi, S. Ito, H. Kobayashi, E. Li, N. Takai, O. Kobayashi, "An Analysis of Stochastic Self-Calibration of TDC Using Two Ring Oscillators", IEEE Asian Test Symposium (Nov. 2013).

[3] William A. Chren Jr., "Low-Area Edge Sampler Using the Chinese Remainder Theorem",IEEE T. Instrumentation and Measurement 48(4): 793-797 (1999).

[4] http://www.ndl.go.jp/math/s1/c2.html

[5] J. Xilinx, San Jose : "Virtex-5 LX FPGA ML501 Evaluation Platform",http://www.xilinx.com/products/boards-and-kits/HW-V5-ML501-UNI-G.htm.

[6] Xilinx, San Jose : "Virtex-5 user guide", 2010. [Online]. Available: www.xilinx.com.

[7] Xilinx, "Using Xilinx ChipScope Pro ILA Core with Project Navigator to Debug FPGA Applications". [Online]. Available: www.xilinx.com.

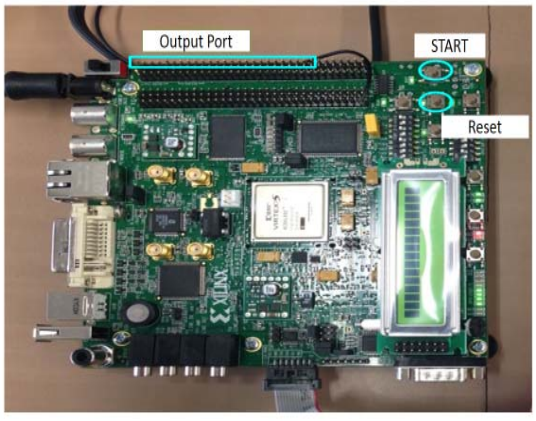




Fig.3 Proposed TDC implementation on FPGA.

Table II Measurement results of the proposed TDC.

| Sample in Window | Elapsed Time (ns) | a | b[0] | b[1] | c[0] | c[1] | c[2] | k |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 30.30 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 6 | 60.60 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 9 | 90.90 | 1 | 0 | 0 | 1 | 1 | 0 | 3 |
| 12 | 121.20 | 0 | 1 | 0 | 0 | 0 | 1 | 4 |
| 15 | 151.50 | 1 | 0 | 1 | 0 | 0 | 0 | 5 |
| 18 | 181.80 | 0 | 0 | 0 | 1 | 0 | 0 | 6 |
| 21 | 212.10 | 1 | 1 | 0 | 0 | 1 | 0 | 7 |
| 24 | 242.40 | 0 | 0 | 1 | 1 | 1 | 0 | 8 |
| 27 | 272.70 | 1 | 0 | 0 | 0 | 0 | 1 | 9 |
| 30 | 303.00 | 0 | 1 | 0 | 0 | 0 | 0 | 10 |
| 33 | 333.30 | 1 | 0 | 1 | 1 | 0 | 0 | 11 |
| 36 | 363.60 | 0 | 0 | 0 | 0 | 1 | 0 | 12 |
| 39 | 393.90 | 1 | 1 | 0 | 1 | 1 | 0 | 13 |
| 42 | 424.20 | 0 | 0 | 1 | 0 | 0 | 1 | 14 |
| 45 | 454.50 | 1 | 0 | 0 | 0 | 0 | 0 | 15 |
| 48 | 484.80 | 0 | 1 | 0 | 1 | 0 | 0 | 16 |
| 51 | 515.10 | 1 | 0 | 1 | 0 | 1 | 0 | 17 |
| 54 | 545.40 | 0 | 0 | 0 | 1 | 1 | 0 | 18 |
| 57 | 575.70 | 1 | 1 | 0 | 0 | 0 | 1 | 19 |
| 60 | 606.00 | 0 | 0 | 1 | 0 | 0 | 0 | 20 |
| 63 | 636.30 | 1 | 0 | 0 | 1 | 0 | 0 | 21 |
| 66 | 666.60 | 0 | 1 | 0 | 0 | 1 | 0 | 22 |
| 69 | 696.90 | 1 | 0 | 1 | 1 | 1 | 0 | 23 |
| 72 | 727.20 | 0 | 0 | 0 | 0 | 0 | 1 | 24 |
| 75 | 757.50 | 1 | 1 | 0 | 0 | 0 | 0 | 25 |
| 78 | 787.80 | 0 | 0 | 1 | 1 | 0 | 0 | 26 |
| 81 | 818.10 | 1 | 0 | 0 | 0 | 1 | 0 | 27 |
| 84 | 848.40 | 0 | 1 | 0 | 1 | 1 | 0 | 28 |
| 87 | 878.70 | 1 | 0 | 1 | 0 | 0 | 1 | 29 |

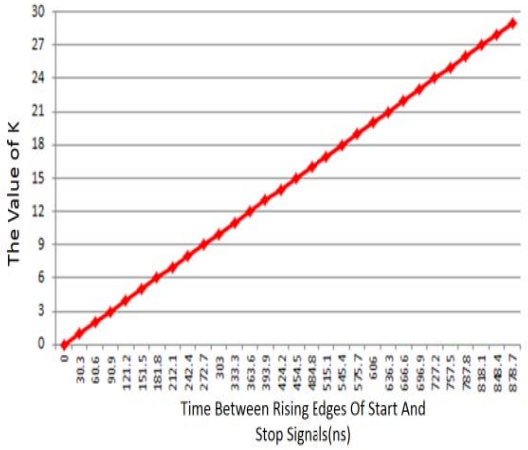



Fig.4 Measurement results of the proposed TDC.

2019 13th IEEE International Conference on ASIC

# Frequency Estimation Sampling Circuit Using Analog Hilbert Filter and Residue Number System

**Yudai Abe**, Shogo Katayama, Congbing Li,
Anna Kuwana, Haruo Kobayashi

Division of Electronics and Informatics
Gunma University

群馬大学
GUNMA UNIVERSITY

Kobayashi Laboratory

# OUTLINE

1. Research Background and Goal
2. Chinese Remainder Theorem
3. Proposed Waveform Sampling Circuit
4. Simulation Verification
5. Summary and Challenge

# OUTLINE

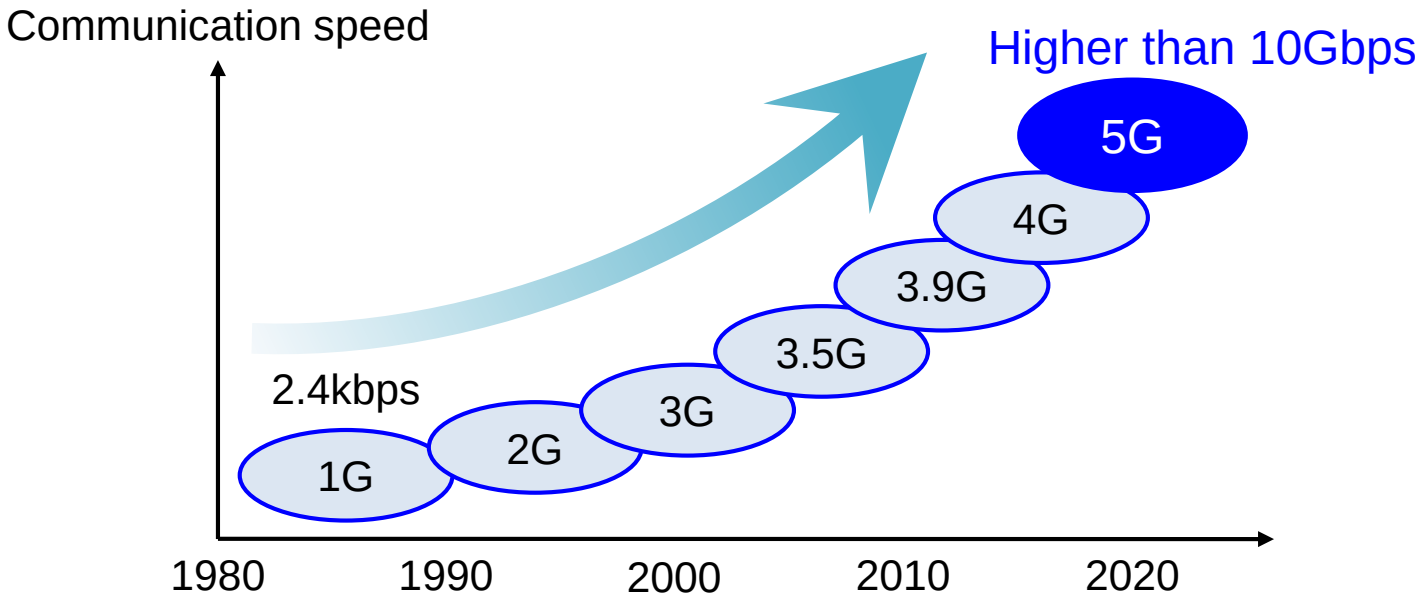
1. Research Background and Goal
2. Chinese Remainder Theorem
3. Proposed Waveform Sampling Circuit
4. Simulation Verification
5. Summary and Challenge

# Research Background

Next Generation Communication System “5G”

High frequencies
in communication systems

Electronic components
for high frequency bands

# Our Research Goal

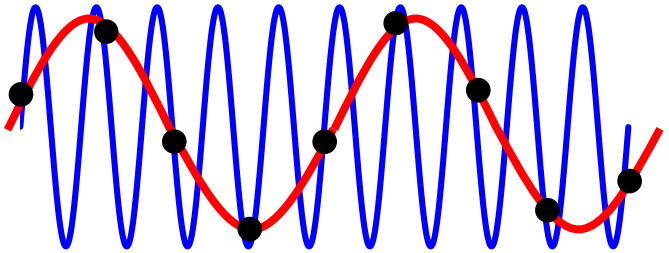
Estimate high-frequency input signal
with multiple low-frequency clock sampling circuits

High-frequency sampling circuit is difficult to realize

Our Approach :

Sampling high frequency signal with multiple low frequency clocks

Use Aliasing proactively



Analog Hilbert filter and residue number system

# OUTLINE

1. Research Background and Goal
2. Chinese Remainder Theorem
3. Proposed Waveform Sampling Circuit
4. Simulation Verification
5. Summary and Challenge

# Chinese Remainder Theorem

Sun Tzu

Chinese arithmetic book ‘Sun Tzu calculation’

孫子算経

“When dividing by 3, its residue is 2,
dividing by 5, its residue is 3,
dividing by 7,its residue is 2.
What is the original number ?”

Answer 23


Sun Tzu calculation

Generalization

Chinese Remainder Theorem

# How to use the Chinese remainder theorem

He used to quickly find out how many soldiers there are.

Sun Tzu

"Divide into 3 people." "Divide into 5 people." How many soldiers are there?"

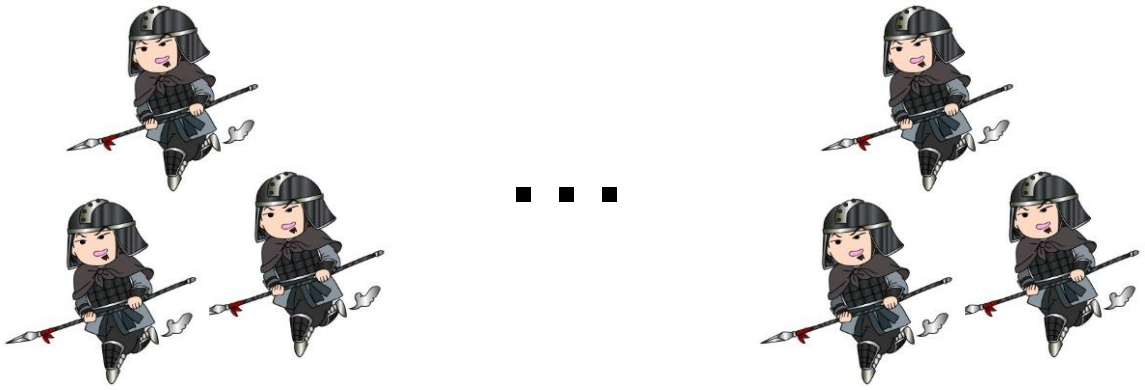
# How to use the Chinese remainder theorem

He used to quickly find out how many soldiers there are.

Sun Tzu

"Divide into 5 people."

Remainder : 2



# How to use the Chinese remainder theorem

He used to quickly find out how many soldiers there are.

Sun Tzu

"Divide into 7 people."

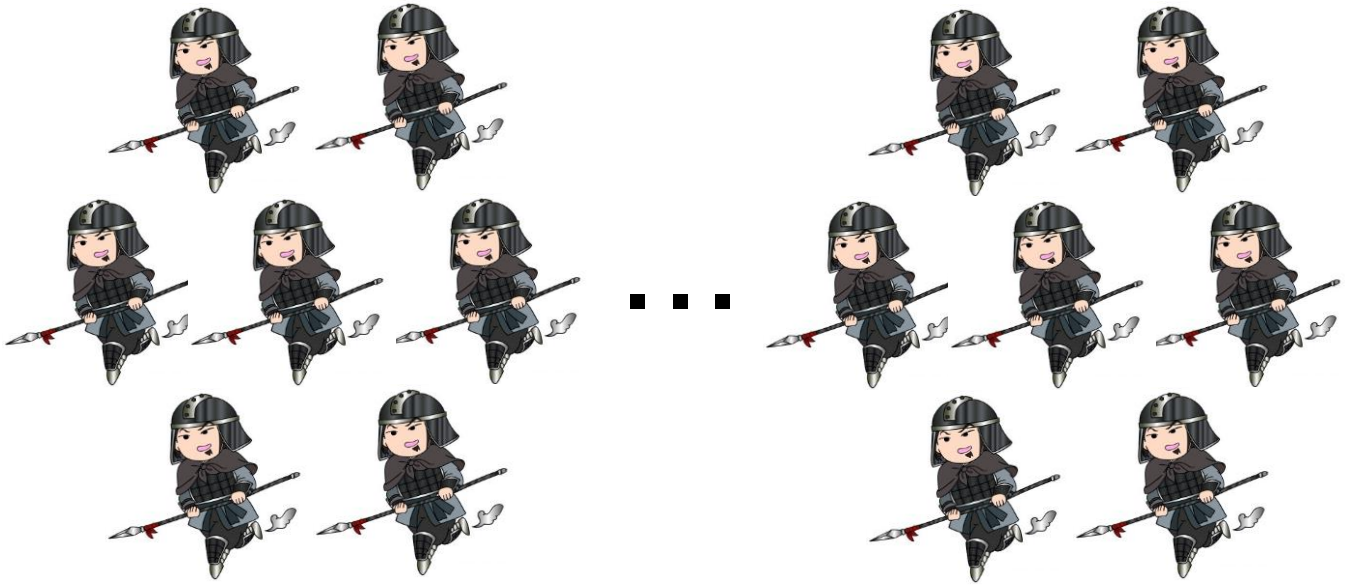
Remainder : 3

# How to use the Chinese remainder theorem

He used to quickly find out how many soldiers there are.

Sun Tzu

"Divide into 7 people."



Remainder : 2

# Example of Residue Number System

23 % 3 = 2,  23 % 5 = 3,  23 % 7 = 2

- Natural numbers
  3, 5, 7 (relatively prime)
  N=3×5×7=105
- k  ( 0 <= k <= N-1 (=104))

a : Remainder of k dividing by 3  a=mod3(k)
b : Remainder of k dividing by 5  b=mod5(k)
c : Remainder of k dividing by 7  c=mod7(k)

k ⟷ (a, b, c)

one to one

Chinese remainder theorem

| a | b | c | k |
|---|---|---|---|
| 0 | 0 | 1 | 15 |
| 1 | 1 | 2 | 16 |
| 2 | 2 | 3 | 17 |
| 0 | 3 | 4 | 18 |
| 1 | 4 | 5 | 19 |
| 2 | 0 | 6 | 20 |
| 0 | 1 | 0 | 21 |
| 1 | 2 | 1 | 22 |
| **2** | **3** | **2** | **23** |
| 0 | 4 | 3 | 24 |
| 1 | 0 | 4 | 25 |
| 2 | 1 | 5 | 26 |
| 0 | 2 | 6 | 27 |
| 1 | 3 | 0 | 28 |
| 2 | 4 | 1 | 29 |

Residue number system

# OUTLINE

1. Research Background and Goal
2. Chinese Remainder Theorem
3. Proposed Waveform Sampling Circuit
4. Simulation Verification
5. Summary and Challenge

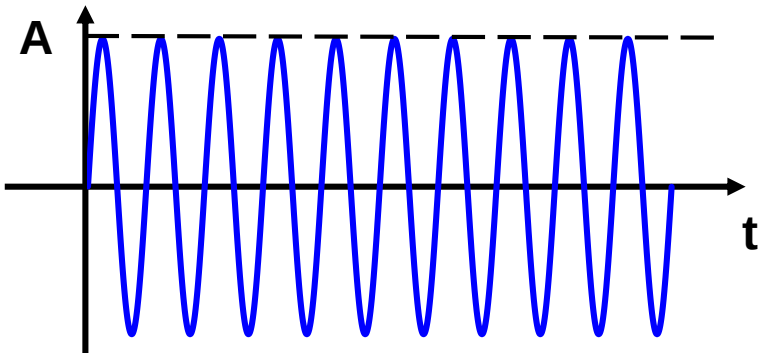
# Aliasing Phenomenon

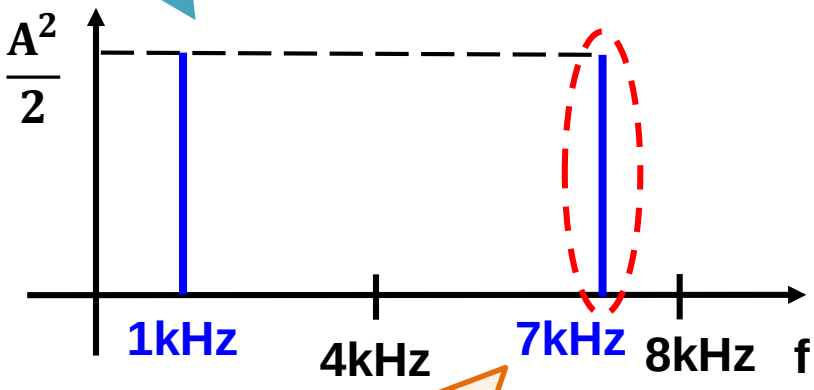
Sampling frequency : 8 kHz

FFT

Spectrums are folded
within the sampling frequency band
( sampling theorem )

Waveform frequency : 31kHz

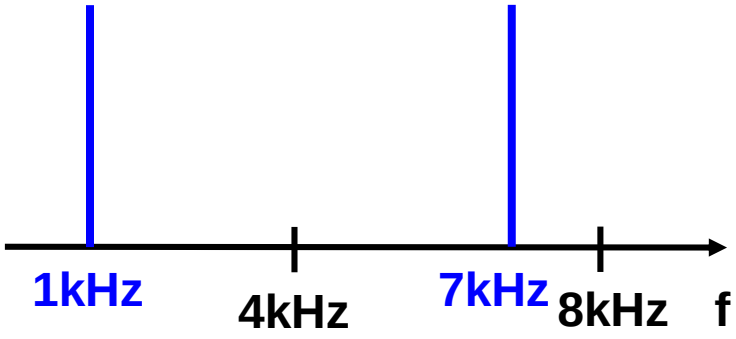


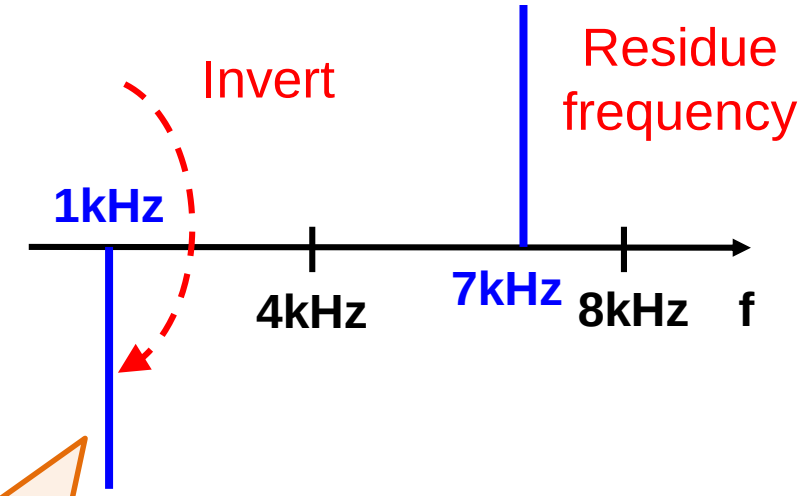

Residue frequency
( 7 is the remainder of 31 divided by 8 )

# Complex FFT of $j \times \sin(2\pi f_{in} t)$

Complex FFT
Input frequency : 31 kHz
Sampling frequency : 8 kHz




$\cos(2\pi f_{in} t)$

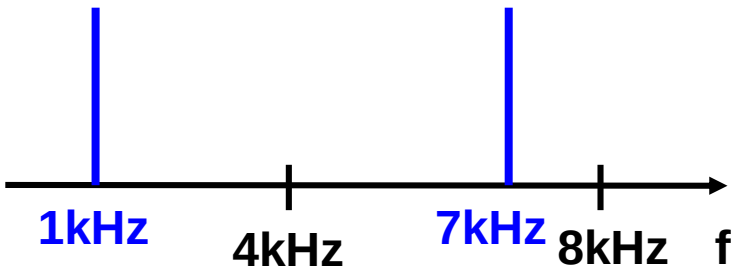



$j \times \sin(2\pi f_{in} t)$

Inverted spectrum
anti-symmetric at Nyquist frequency

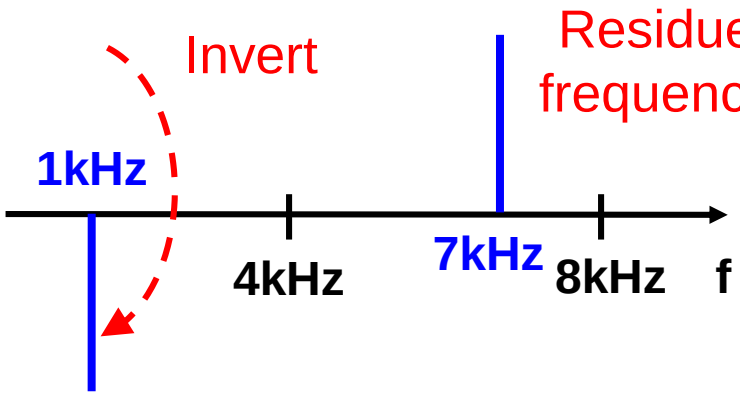
# Complex FFT of $\cos(2\pi f_{in}t) + j \times \sin(2\pi f_{in}t)$

Complex FFT
Input frequency : 31 kHz
Sampling frequency : 8 kHz

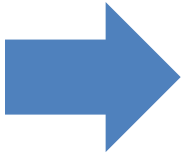


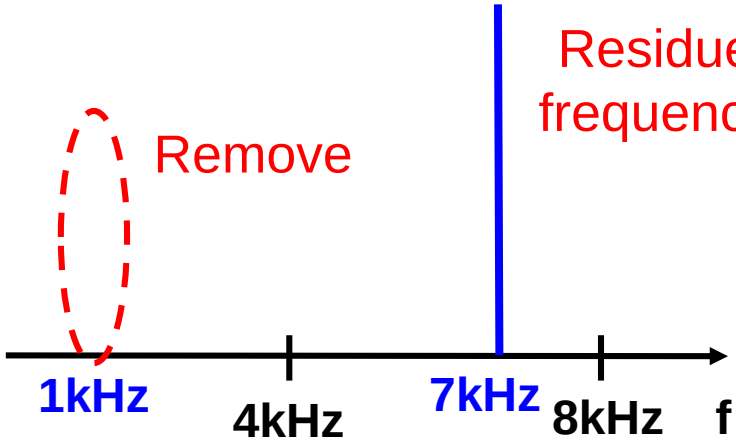

$\cos(2\pi f_{in}t)$

+




$j \times \sin(2\pi f_{in}t)$






$\cos(2\pi f_{in}t) + j \times \sin(2\pi f_{in}t)$

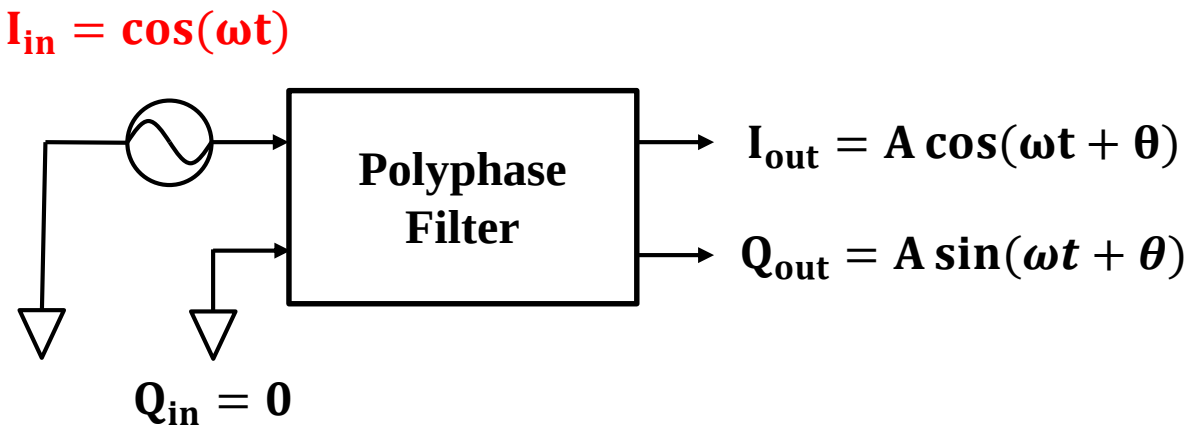
Extract spectrum
of the residual frequency

# How Generate $j \times \sin(2\pi f_{in} t)$

Use Analog Hilbert filter

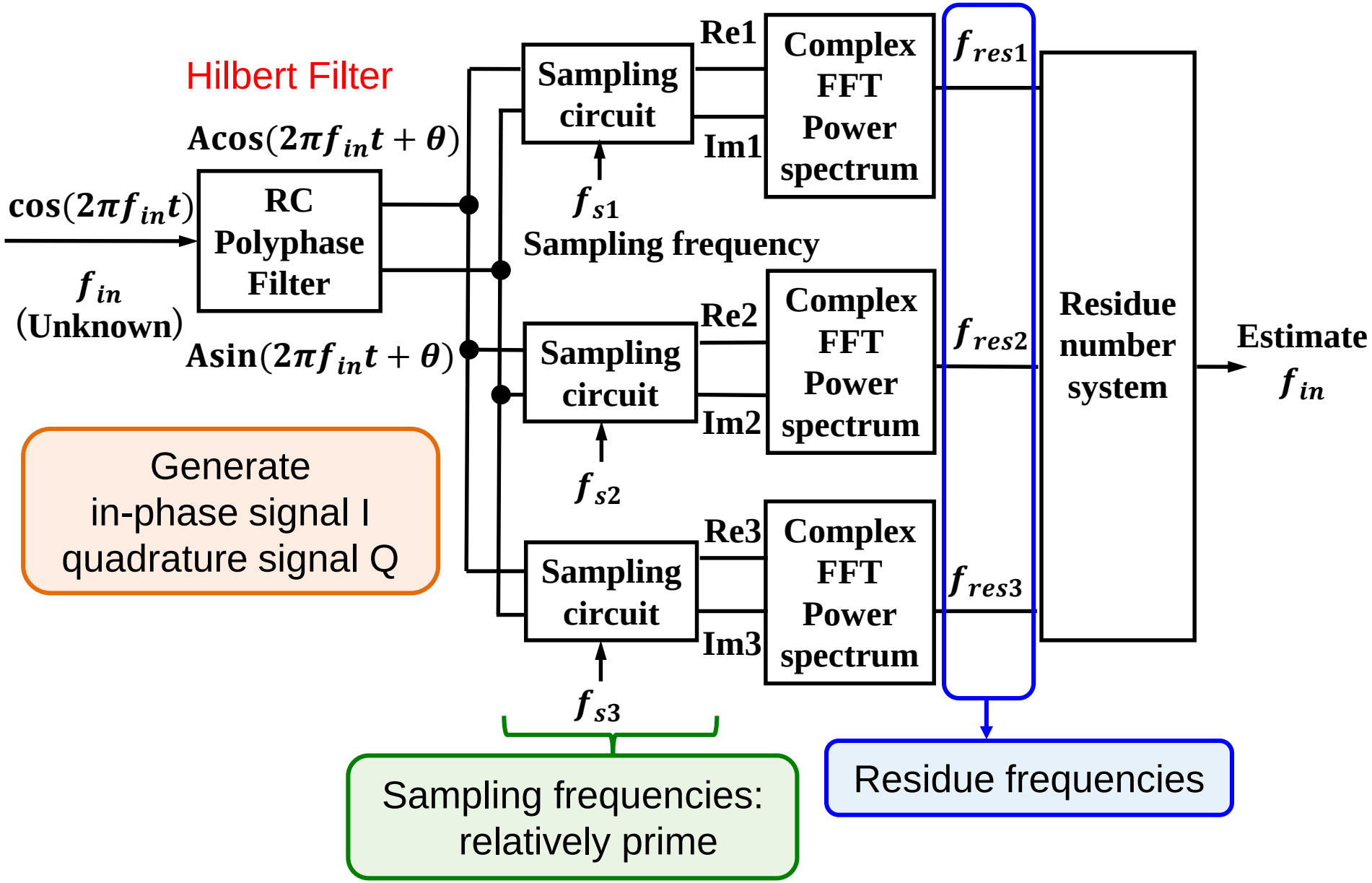
RC polyphase filter

David Hilbert
(German mathematician)
1862-1943




Generate in-phase and quadrature waves from a single cosine wave

# Proposed Sampling Circuit



# OUTLINE

1. Research Background and Goal
2. Chinese Remainder Theorem
3. Proposed Waveform Sampling Circuit
4. Simulation Verification
5. Summary and Challenge

# Simulation Settings

**Complex FFT**

- Input frequency : 12 GHz
- Frequency resolution : 1 kHz
- Sampling frequency : 229 kHz, 233 kHz, 239 kHz
  ( Relatively prime )
- Range of measurement : 0~2080622 kHz
  ( Note: 229 × 233 × 239 = 2080623 )

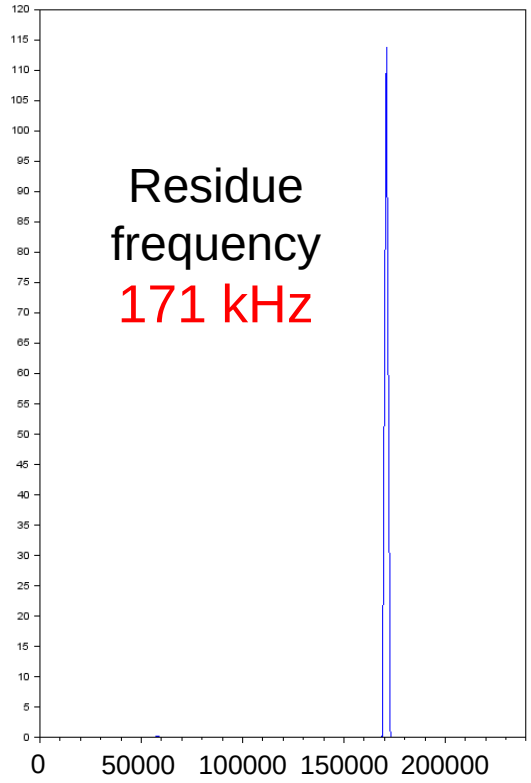
Measurement at 20 GHz
using sampling frequencies of ≒ 200 kHz

# Simulation Results

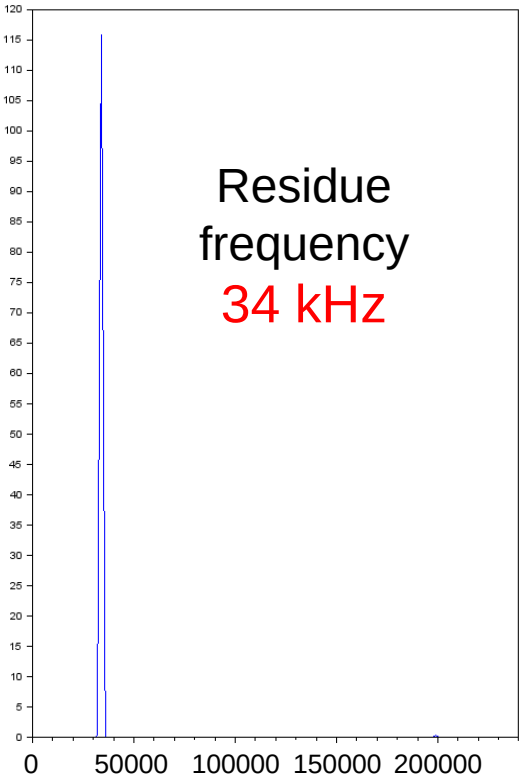
Complex FFT : $\cos(2\pi f_{in}t) + j \times \sin(2\pi f_{in}t)$

- Input frequency : 12 GHz
- Frequency resolution : 1 kHz
- Sampling frequency : 229 kHz  233 kHz  239 kHz

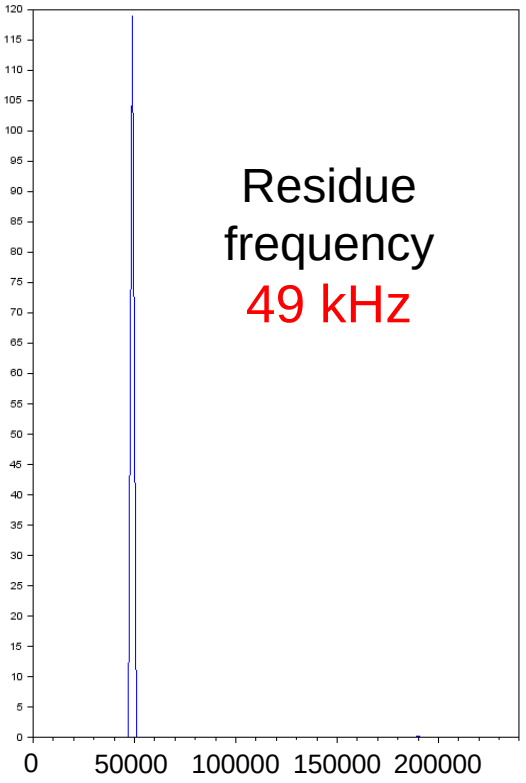
229 kHz Sampling




233 kHz Sampling




239 kHz Sampling



# Frequency Estimation by Residue Number System

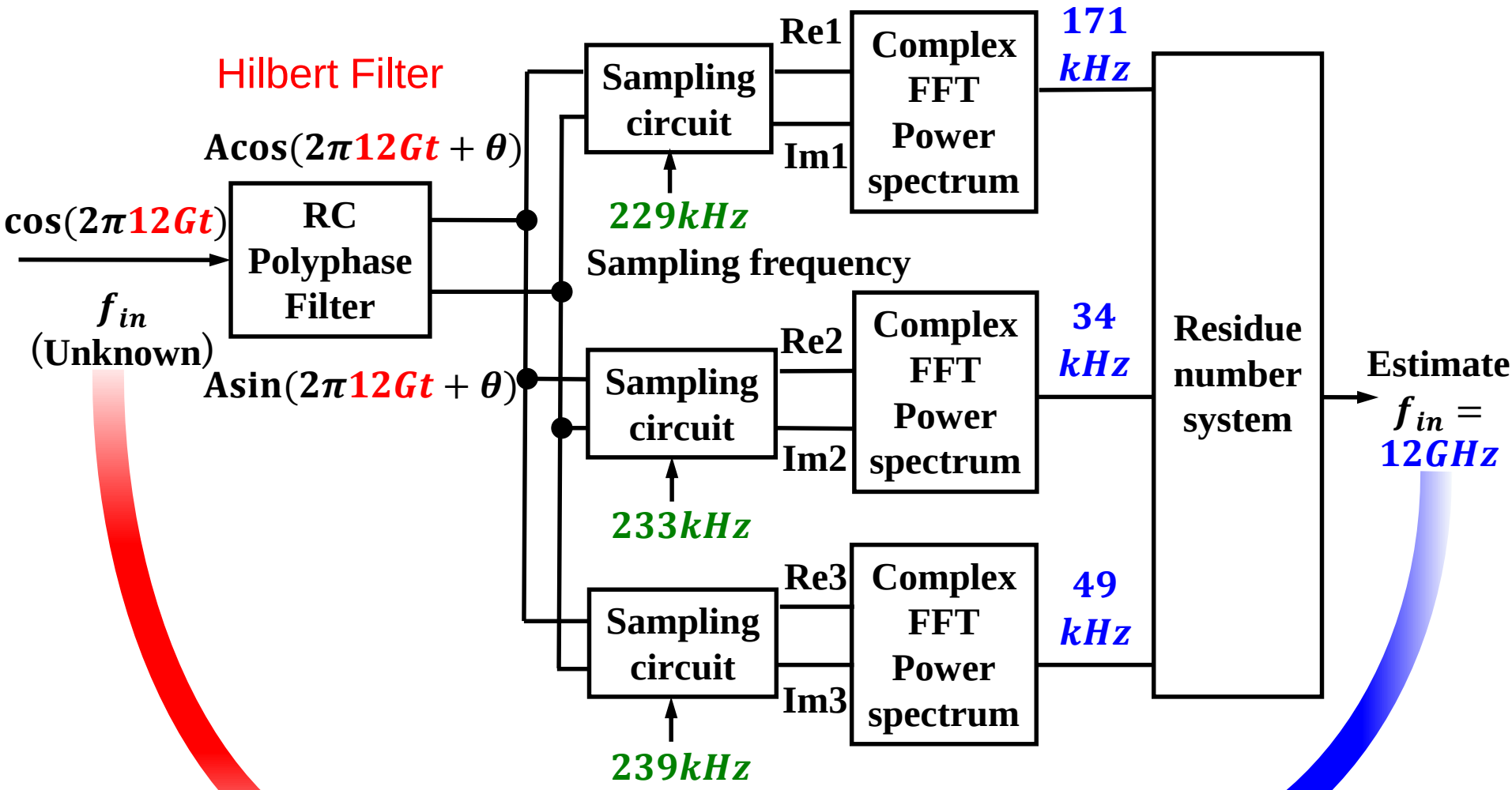
Residue frequencies
171 kHz, 34 kHz, 49 kHz

Input frequency estimation
using residue frequencies
and residue number system

Estimate input frequency 12GHz

| a [kHz] | b [kHz] | c [kHz] | k [kHz] |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 169 | [illegible] | 47 | 11999998 |
| 170 | [illegible]3 | 48 | 11999999 |
| 171 | 34 | 49 | 12000000 |
| 172 | 35 | 50 | 120[illegible]0001 |
| 173 | 36 | 51 | 12[illegible]00002 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 226 | 230 | [illegible] | 12752320 |
| [illegible] | [illegible]31 | 237 | 12752321 |
| 228 | 232 | 238 | 12752322 |

# Simulation Result Overview



# OUTLINE

1. Research Background and Goal
2. Chinese Remainder Theorem
3. Proposed Waveform Sampling Circuit
4. Simulation Verification
5. Summary and Challenge

# Summary and Challenge

## Summary

- Proposed a method to estimate high-frequency signal using multiple low-frequency sampling circuits.
- Confirmed its operation by theory and simulation.
- Measurable range is wide:
  proportional to multiplication of multiple sampling frequencies.

## Challenge

- Estimated input frequency is discrete

→ Consider estimation with fine frequency resolution

Thank you for your attention

2015 IEEE International Symposium on
Radio-Frequency Integration Technology (RFIT2015)
August 26-28, 2015
Sendai, Japan

# A Gray Code Based Time-to-Digital Converter Architecture and its FPGA Implementation

**Congbing Li  Haruo Kobayashi**

**Gunma University**

# Outline

- Research Objective & Background
- Flash TDC and Problems
- Gray Code
- Gray Code TDC Architecture
- FPGA Implementation
- RTL Verification of Glitch-free Characteristic
- Conclusion

# Research Objective

## Objective

- Development of

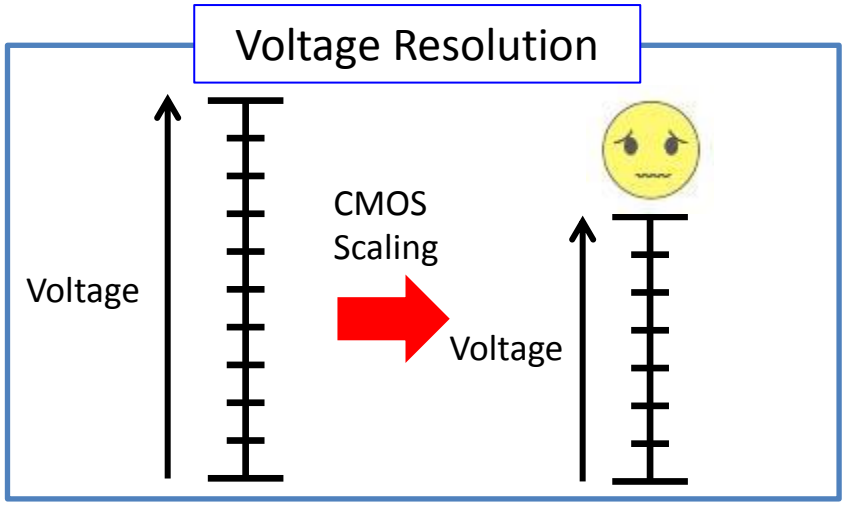
  Time-to-Digital Converter (TDC) architecture with high-speed and small hardware

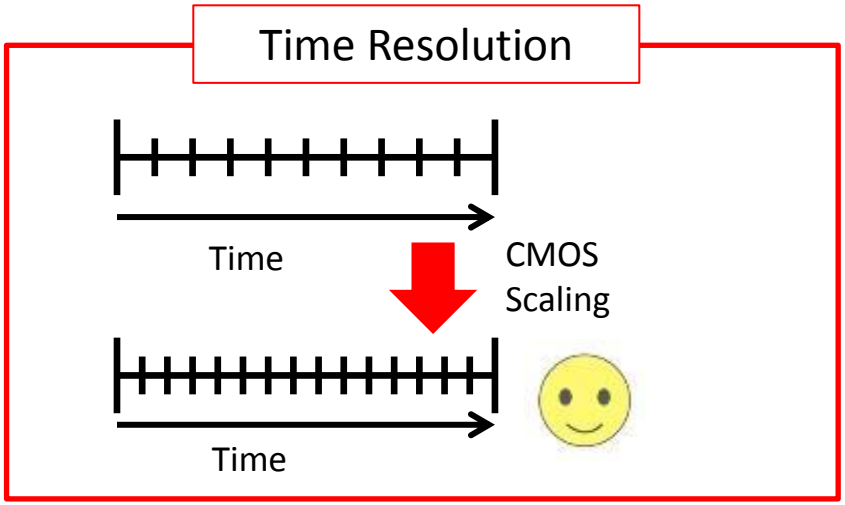
## Approach

- Utilization of Gray code

# Research Background

**TDC plays an important role in nano-CMOS era**



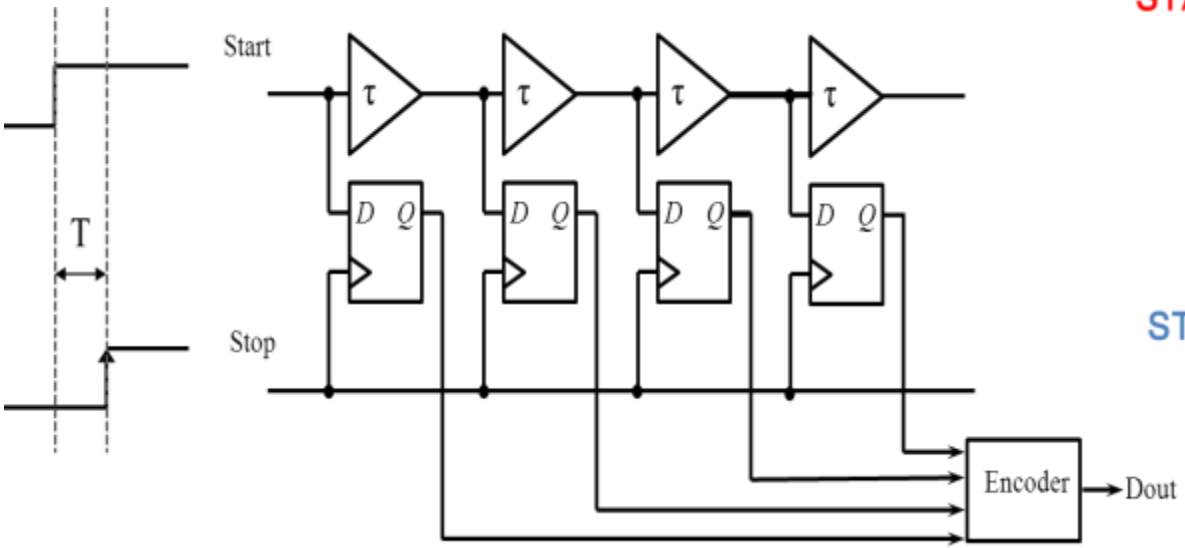



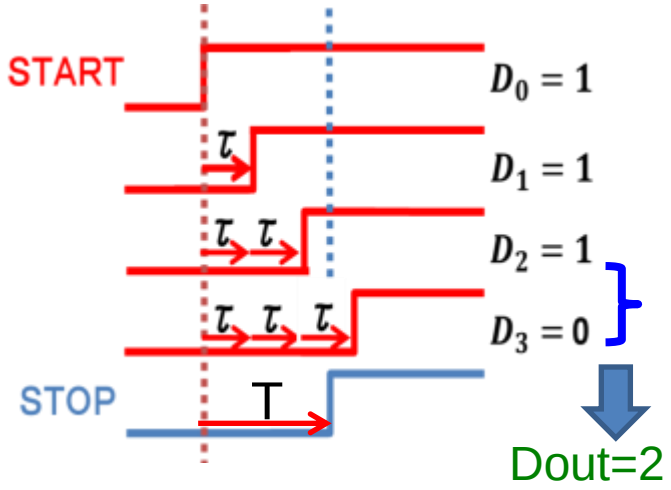

Voltage-domain resolution facing difficulties due to reduced supply voltage

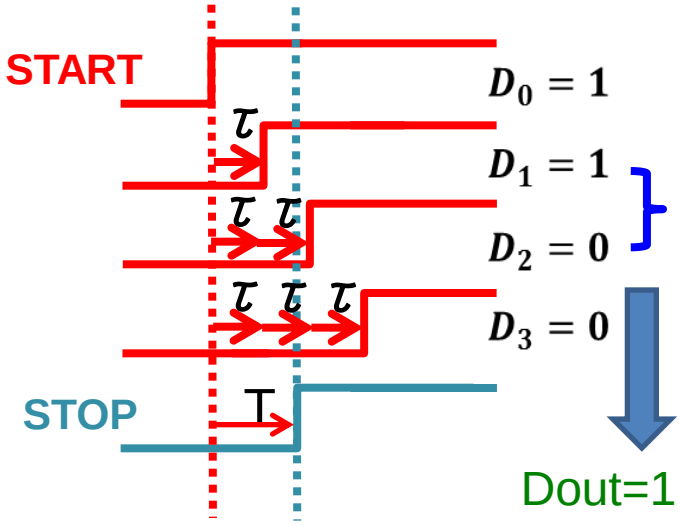
Time-domain resolution becoming superior

TDC measures time interval between two signal transitions, into digital signal.
(widely used in ADPLLs, jitter measurements, time-domain ADC)

# Flash TDC




- Digital output (Dout) proportional to time difference between rising edges (T)
- Time resolution $\tau$

# Problems of Flash TDC

An n-bit flash TDC with $2^n$ quantization levels

Advantages

- High-speed timing measurement
- Single-event timing measurement
- All digital implementation

Disadvantages

- $2^n-1$ delay elements, $2^n-1$ Flip-Flops
- n-bit thermometer-to-binary code encoder

Large circuits
High power consumption

# Gray Code (1/2)

Gray Code

a binary numeral system where two successive values differ in only one bit

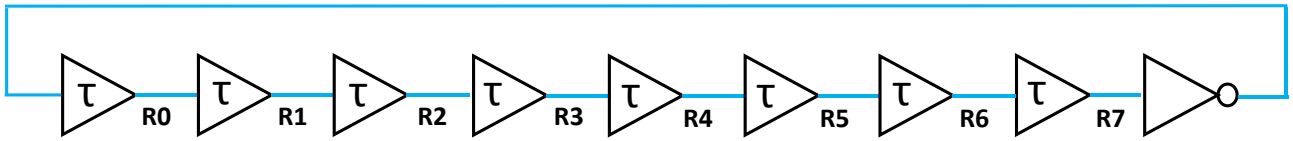
**Table. 4-bit Gray Code**

| Decimal numbers | Binary Code | Gray Code |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

- For Gray code, between any two adjacent numbers, only one bit changes at a time
- Gray code data is more reliable compared with binary code

# Gray Code (2/2)

In a ring oscillator, between any two adjacent states, only one output changes at a time. This characteristic is very similar to Gray code.




| 8-stage Ring Oscillator Output | | | | | | | | 4-bit Gray Code | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| R0 | R1 | R2 | R3 | R4 | R5 | R6 | R7 | G3 | G2 | G1 | G0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

For any given Gray code, its each bit can be generated by a certain ring oscillator.

# Gray Code based TDC Architecture (1/2)

**A Gray code TDC architecture can be conceived by grouping a few ring oscillators**

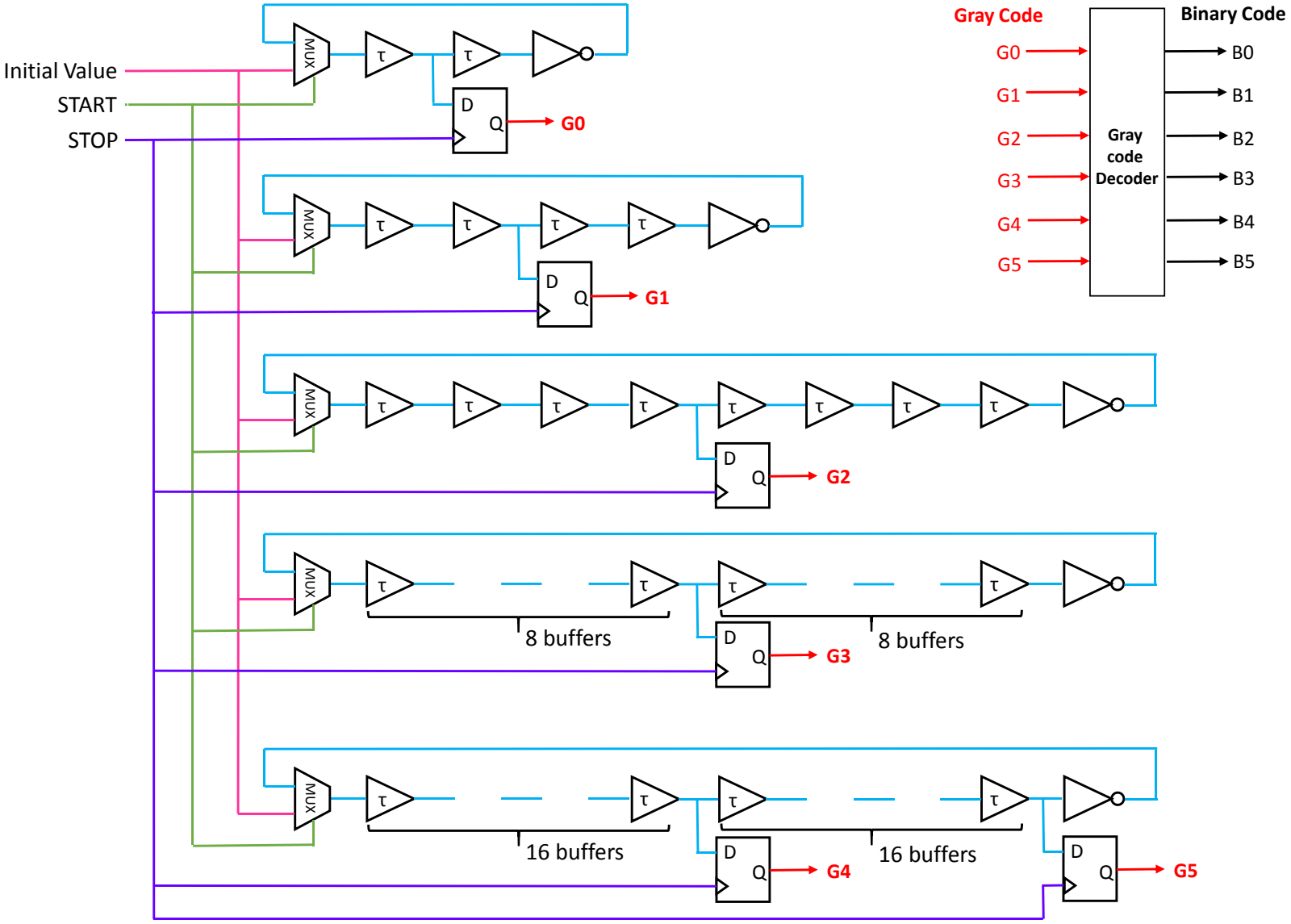
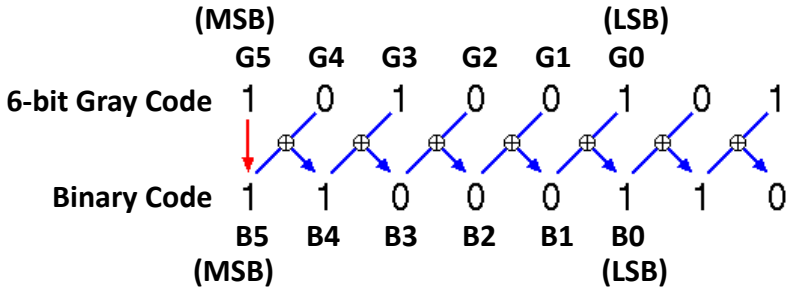



**Figure. Proposed 6-bit Gray code TDC**

# Gray Code based TDC Architecture (2/2)




**B5=G5**
**B4=B5 ⊕ G4**
**B3=B4 ⊕ G3**
**B2=B3 ⊕ G2**
**B1=B2 ⊕ G1**
**B0=B1 ⊕ G0**

**Figure. Gray code decoder**

## Flash vs. Proposed TDCs

for a measurement range of $2^6$

| | Flash TDC | Proposed TDC |
|---|---|---|
| Number of delay elements | 64 | 62 |
| Number of Flip-flop | 64 | 6 |
| The maximum stage | 64 | 32 |

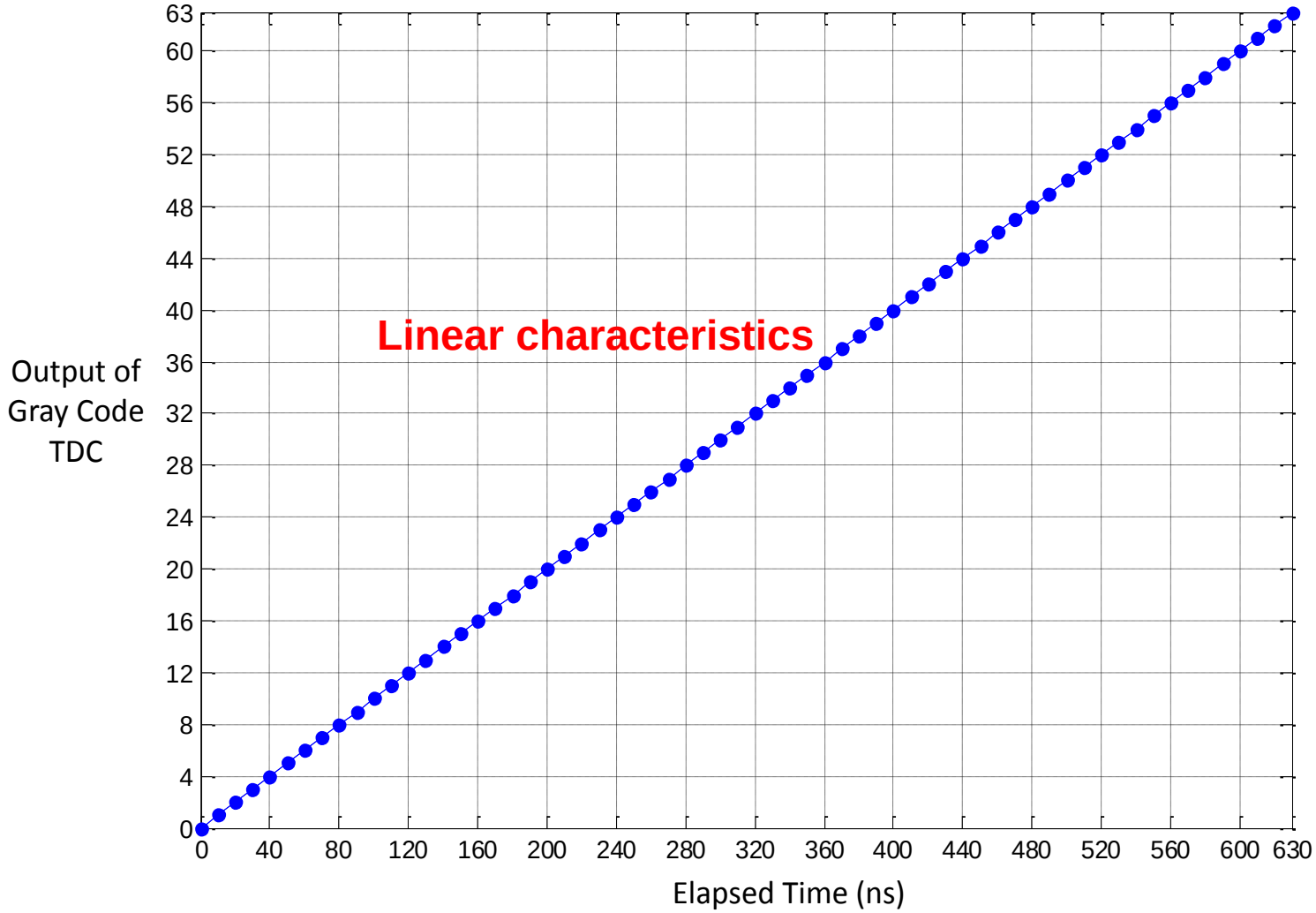
for a measurement range of $2^n$ significant hardware reduction as # of bits increases.

# FPGA Implementation (1/3)

Proposed TDC implementation on Xilinx FPGA


Note: ADC is difficult to implement with full digital FPGA.

# FPGA Implementation (2/3)

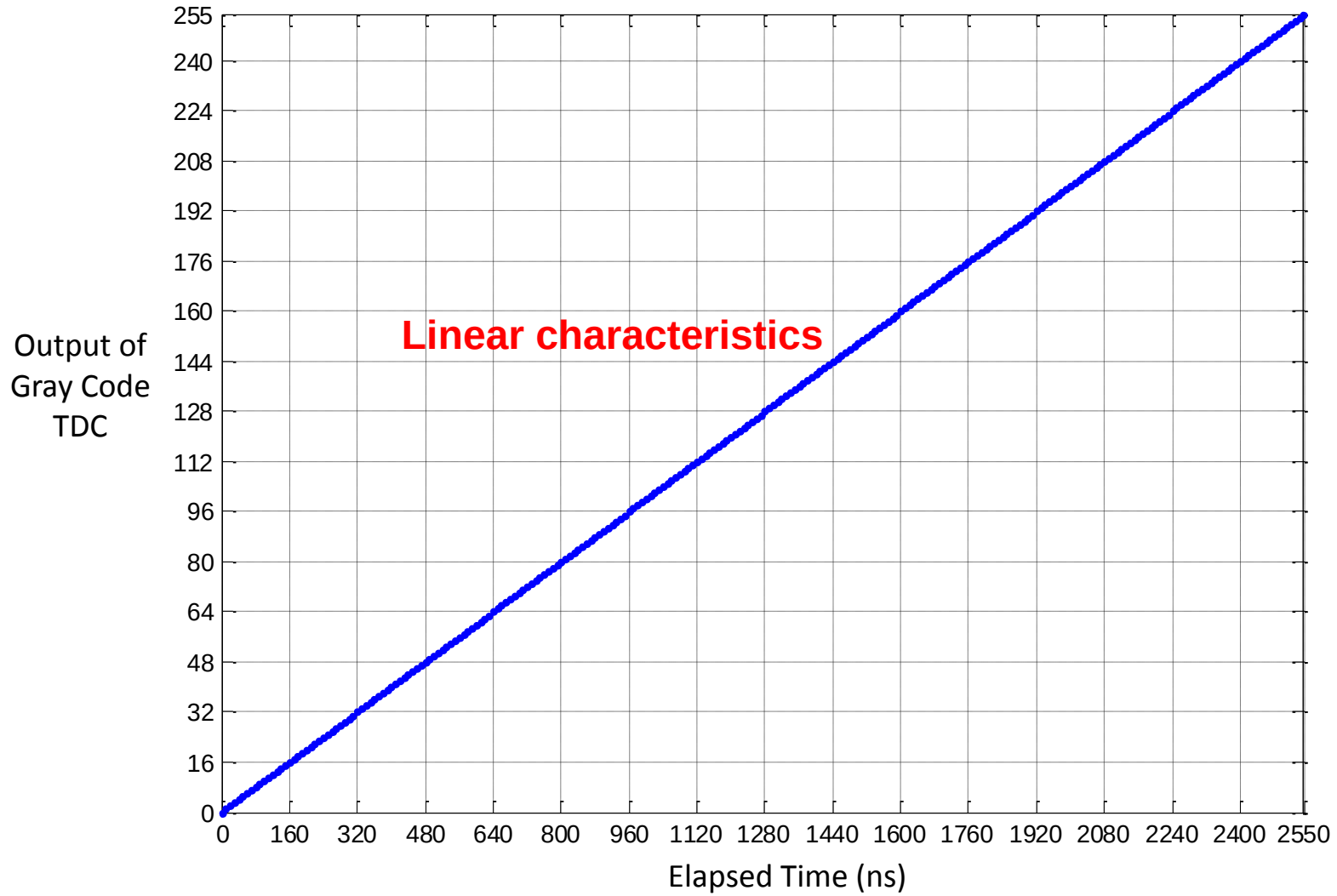



**Measurement results of the proposed TDC with FPGA (6-bit case).**

**Proposed TDC operation is confirmed with FPGA evaluation.**

# FPGA Implementation (3/3)

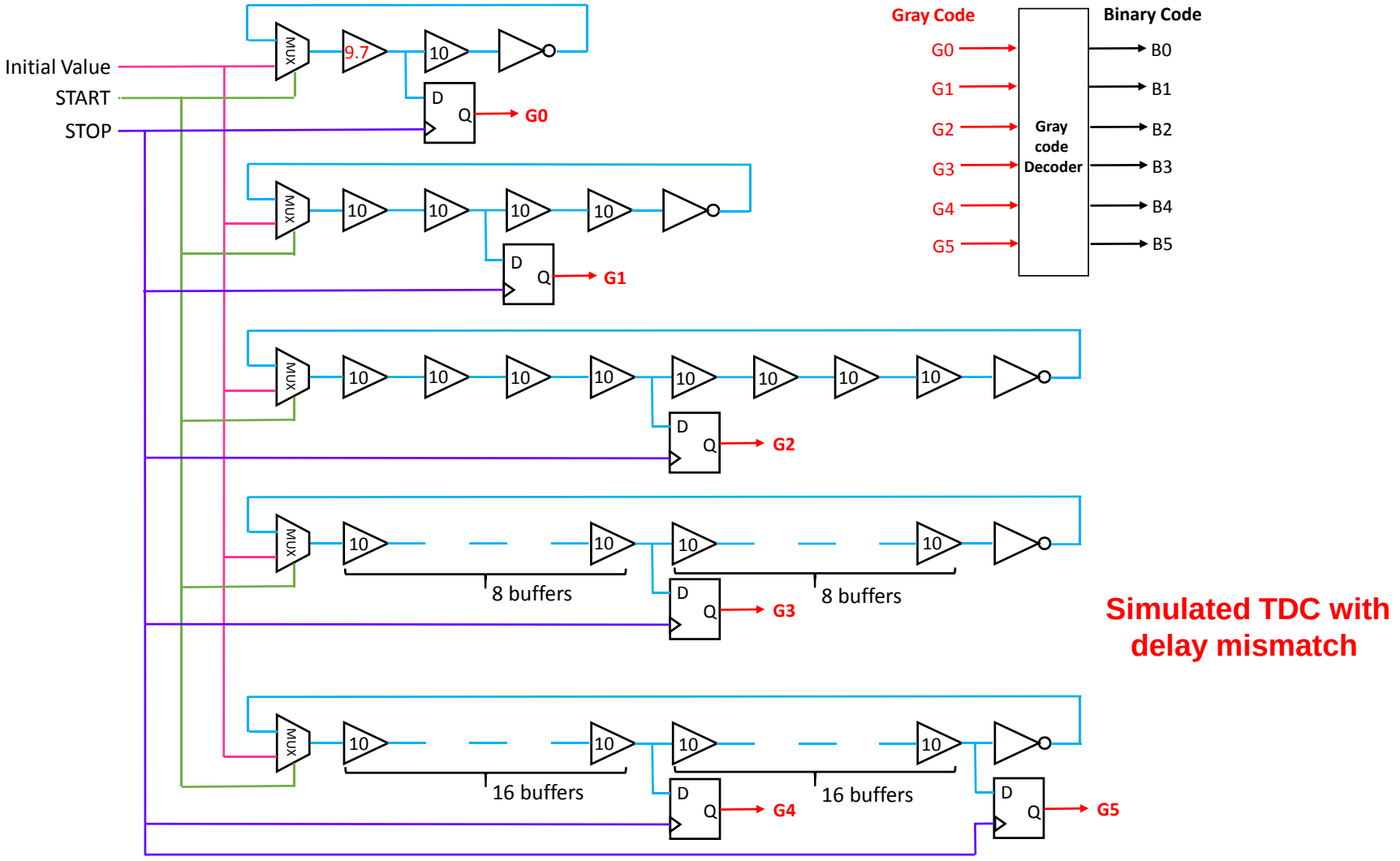
**Similarly, 8-bit Gray code TDC architecture was implemented on FPGA.**




**Measurement results of the proposed TDC with FPGA (8-bit case)**

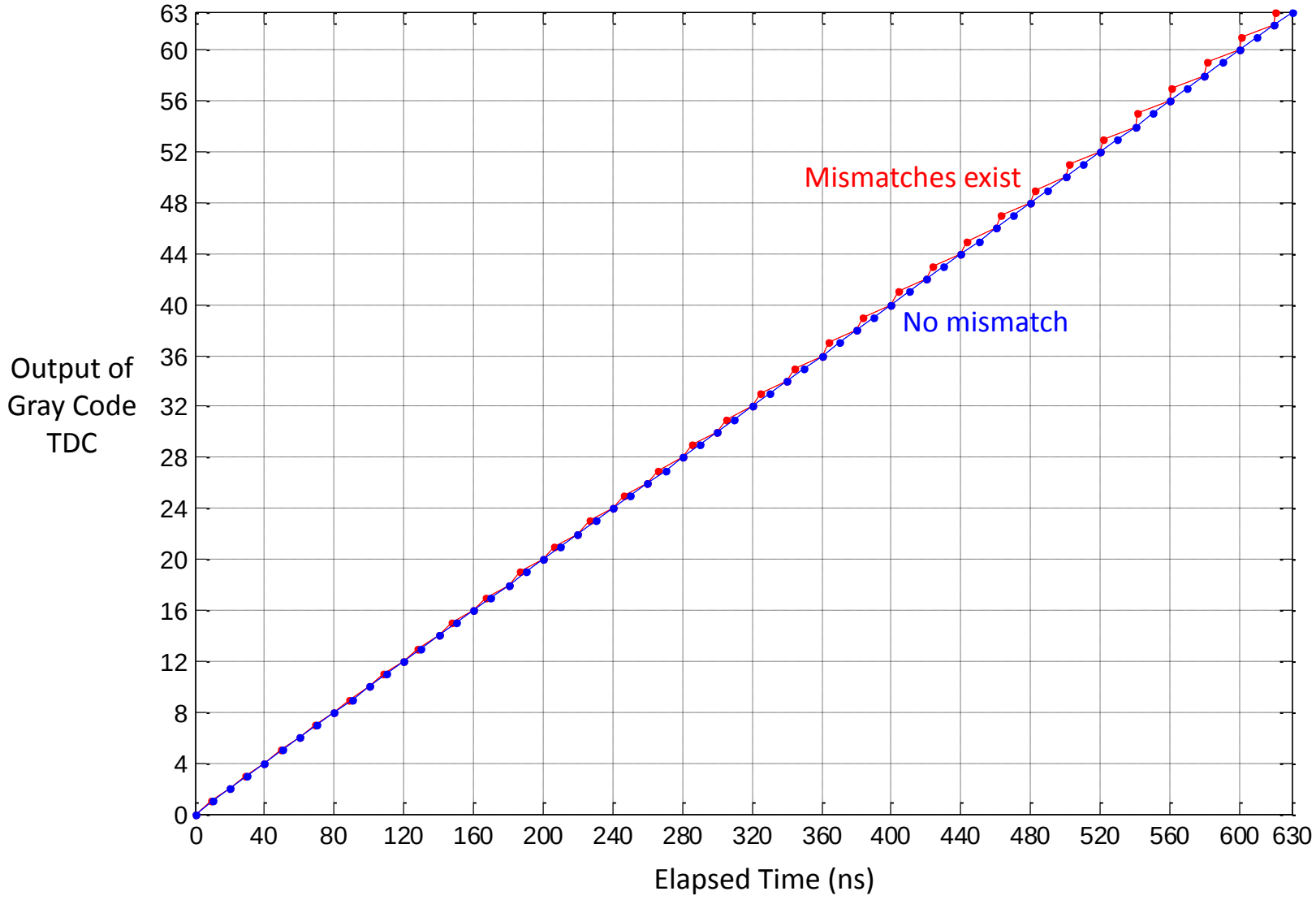
# RTL Verification of Glitch-free Characteristic (1/2)

- The proposed Gray code TDC can provide a glitch-free binary code sequence even there are mismatches between the delay stages.
- RTL simulation was conducted to verify this characteristic.



# RTL Verification of Glitch-free Characteristic (2/2)

- RTL simulation result shows that no matter there are mismatches among the delay stages or not, the proposed Gray code TDC can always output a glitch-free binary code sequence.



# Conclusion

We have proposed a gray code based TDC architecture

- Comparable performance to Flash TDC
- Significant hardware & power reduction

for a measurement range of $2^n$

| | Flash TDC | Proposed TDC |
|---|---|---|
| Number of delay elements | $2^n$ | $2^n - 2$ |
| Number of Flip-flop | $2^n$ | $n$ |
| The maximum stage | $2^n$ | $2^{n-1}$ |

Significant hardware reduction as # of bits increases.

We have implemented the proposed TDC with FPGA

➡ Confirmed its operation

IEEE International Symposium on Intelligent Signal Processing and Communication Systems 2017

NOVEMBER 6-9, 2017, XIAMEN, CHINA

Nov. 9 NA-L2 8:30-9:50

# Gray-Code Input DAC Architecture for Clean Signal Generation

Richen.Jiang, G.Adhikari, Yifei.Sun, Dan.Yao,

R.Takahashi, Y.Ozawa, N.Tsukiji, H.Kobayashi, R.Shiota

*Gunma University*, *Socionext Inc.*,

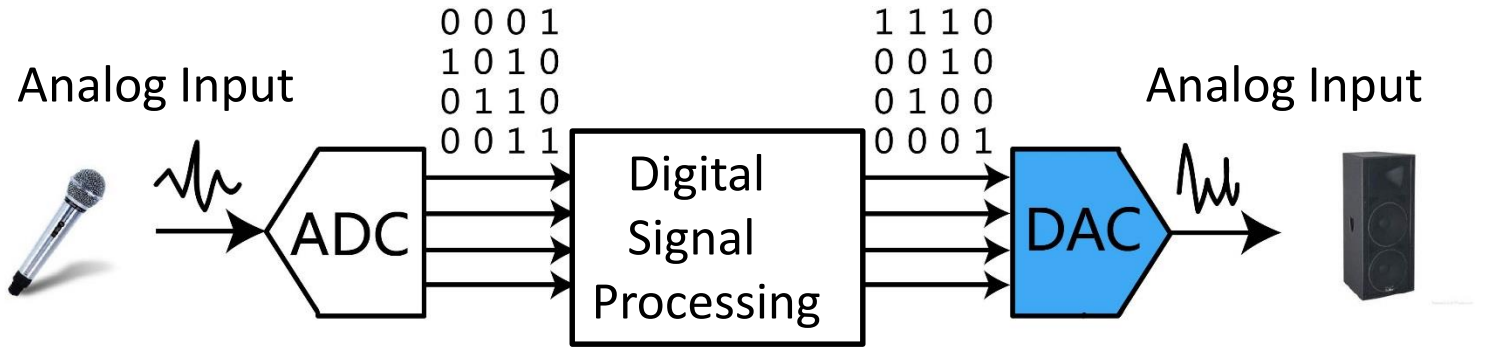
# OUTLINE

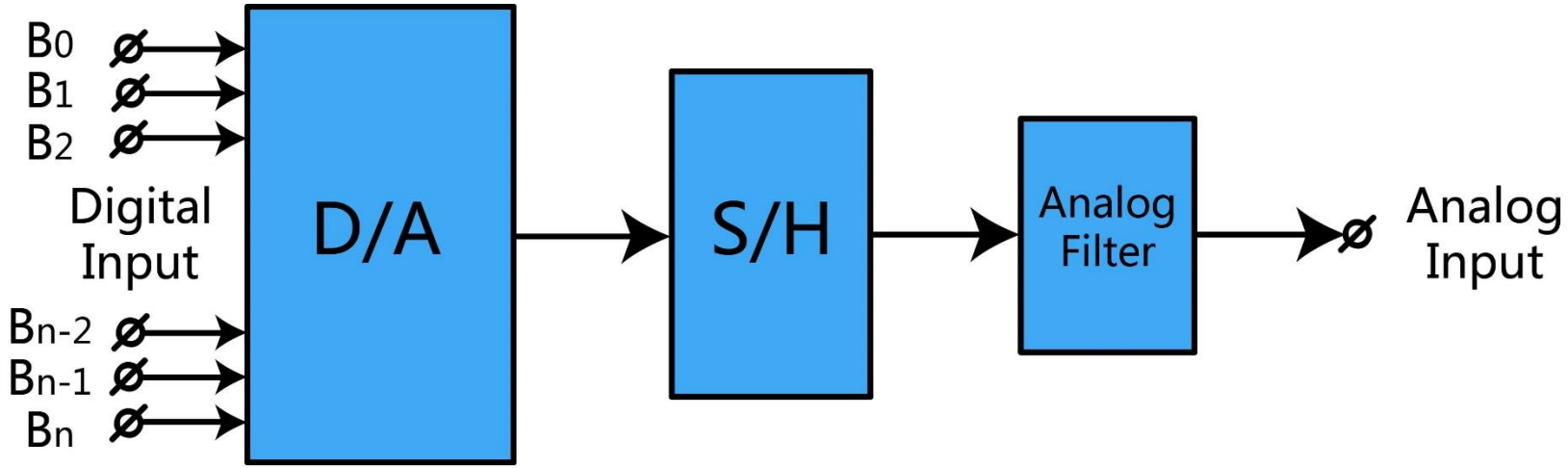
- Research Background・Objective
- Glitches
- Gray-code
- Gray-code Input DAC Architecture and Operation
- Simulation Verification by SPICE
- Conclusion

# OUTLINE

- Research Background・Objective
- What are Glitches
- Gray-code
- Gray-code Input DAC Architecture and Operation
- Simulation Verification by SPICE
- Conclusion

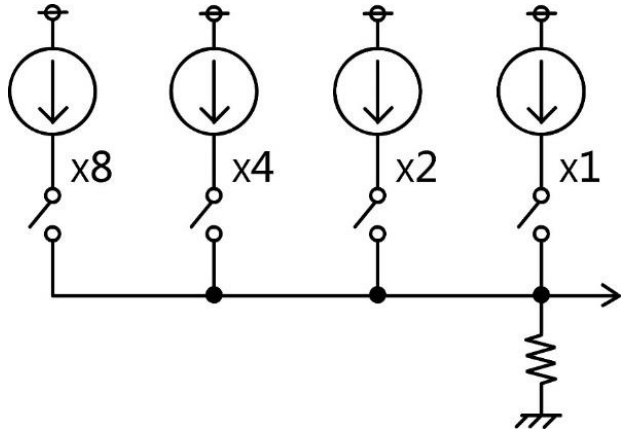
# Research Background



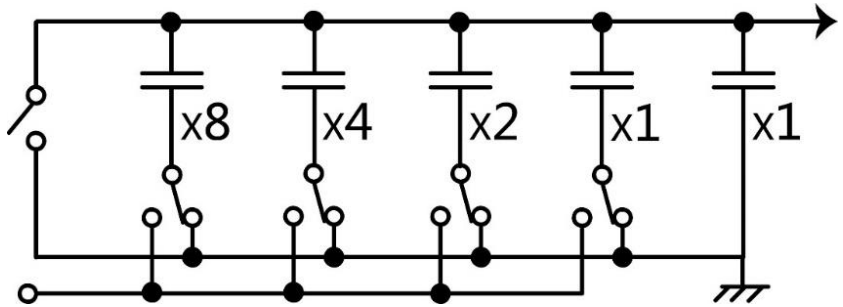



# Research Background

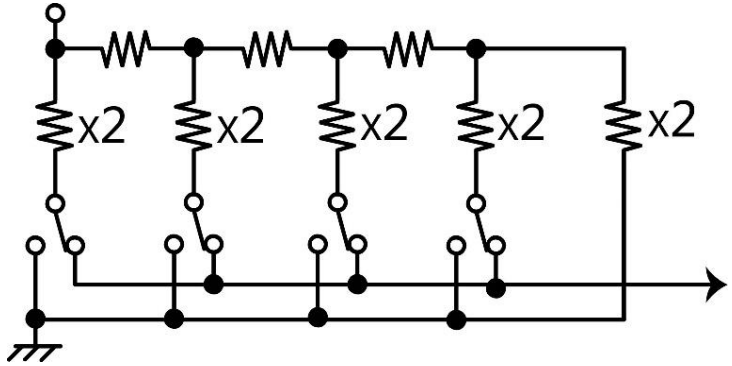
Basic architecture of DAC




Current Source DAC




Capacitive DAC




Resistance DAC

The switch is driven with a binary code → glitch

# Research Objective

## Objective

- Design Digital-to-Analog Converter (DAC) architectures for clean signal generation

## Approach

- By reducing glitches with Gray-Code input topologies

# OUTLINE

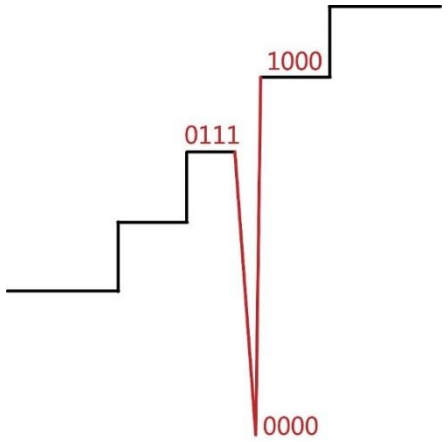
- Research Background・Objective
- Glitches
- Gray-code
- Gray-code Input DAC Architecture and Operation
- Simulation Verification by SPICE
- Conclusion

# What are Glitches

- Voltage spikes
- Reasons for glitches

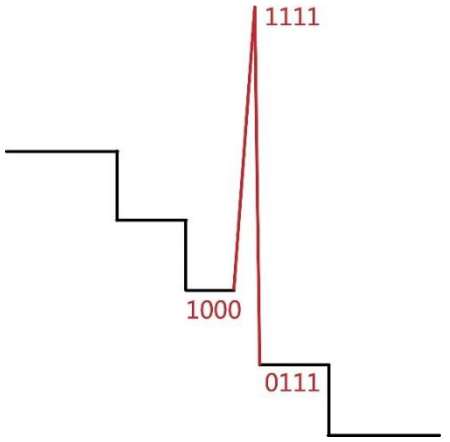
| Decimal numbers | Natural Binary code |
|---|---|
| 0 | 0 0 0 0 |
| 1 | 0 0 0 1 |
| 2 | 0 0 1 0 |
| 3 | 0 0 1 1 |
| 4 | 0 1 0 0 |
| 5 | 0 1 0 1 |
| 6 | 0 1 1 0 |
| 7 | 0 1 1 1 |
| 8 | 1 0 0 0 |
| 9 | 1 0 0 1 |
| 10 | 1 0 1 0 |
| 11 | 1 0 1 1 |
| 12 | 1 1 0 0 |
| 13 | 1 1 0 1 |
| 14 | 1 1 1 0 |
| 15 | 1 1 1 1 |

when 7→8




0111→0110→0100→**0000**→1000

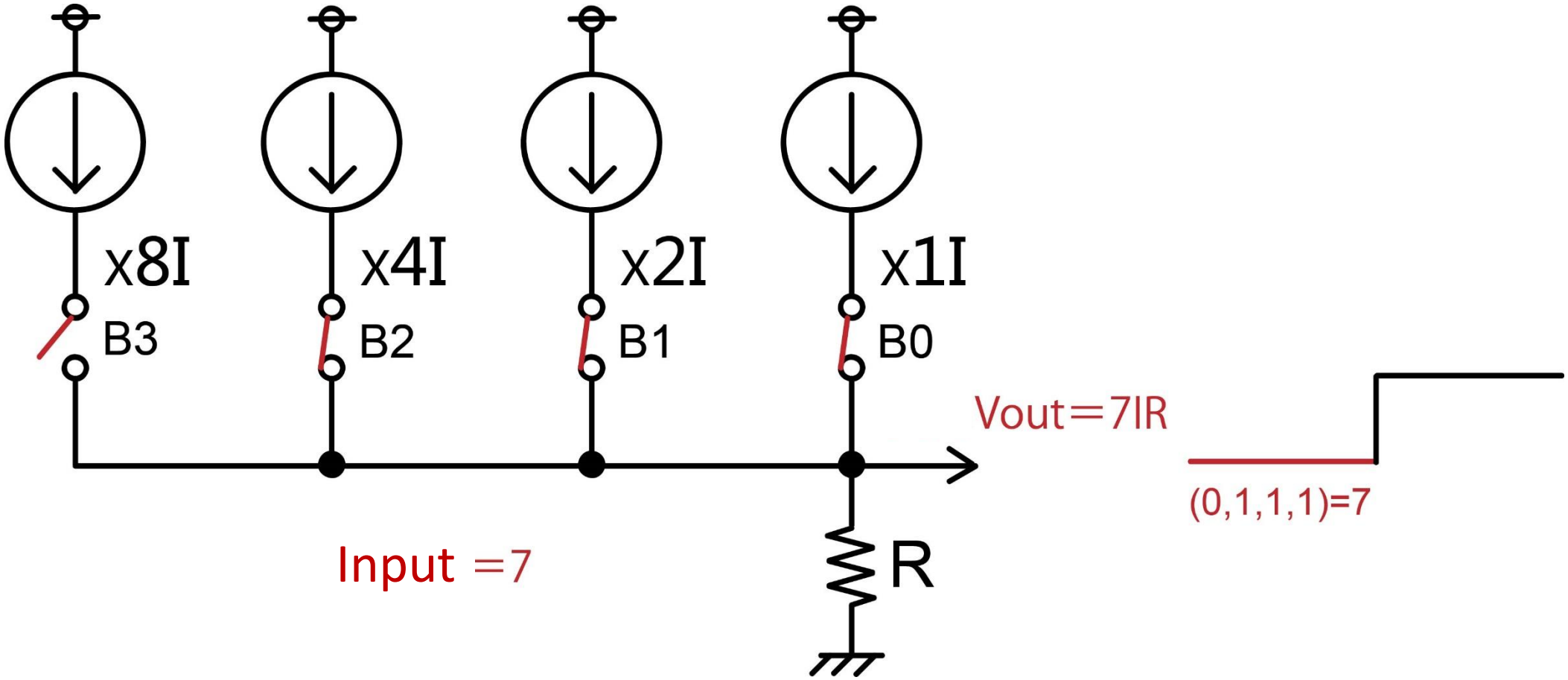
when 8→7




1000→1001→1011→**1111**→0111

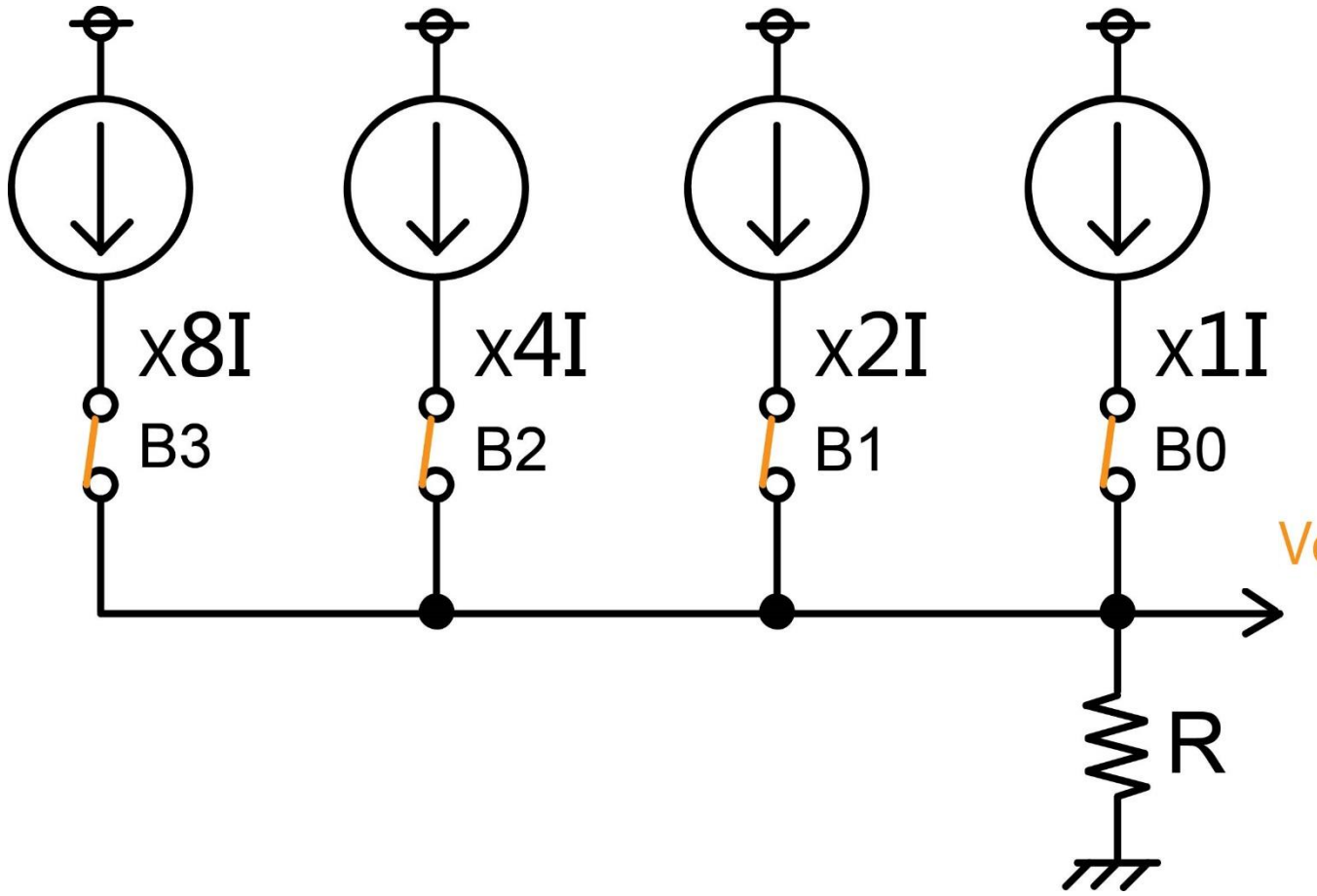
The most significant bit (MSB) changes (near the middle point)

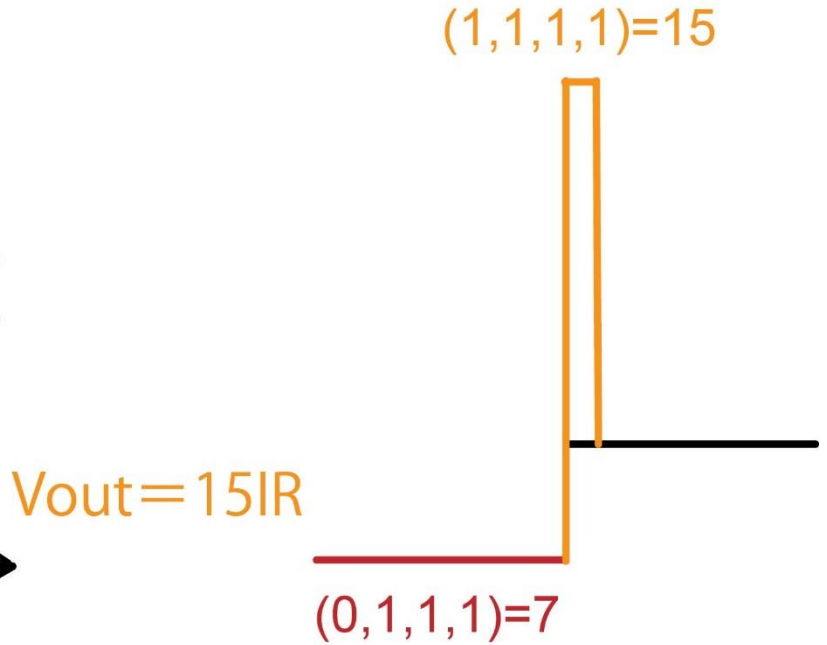
# Generation of Glitch at Switching time




When the input changes 7 →8

# Generation of Glitch at Switching time

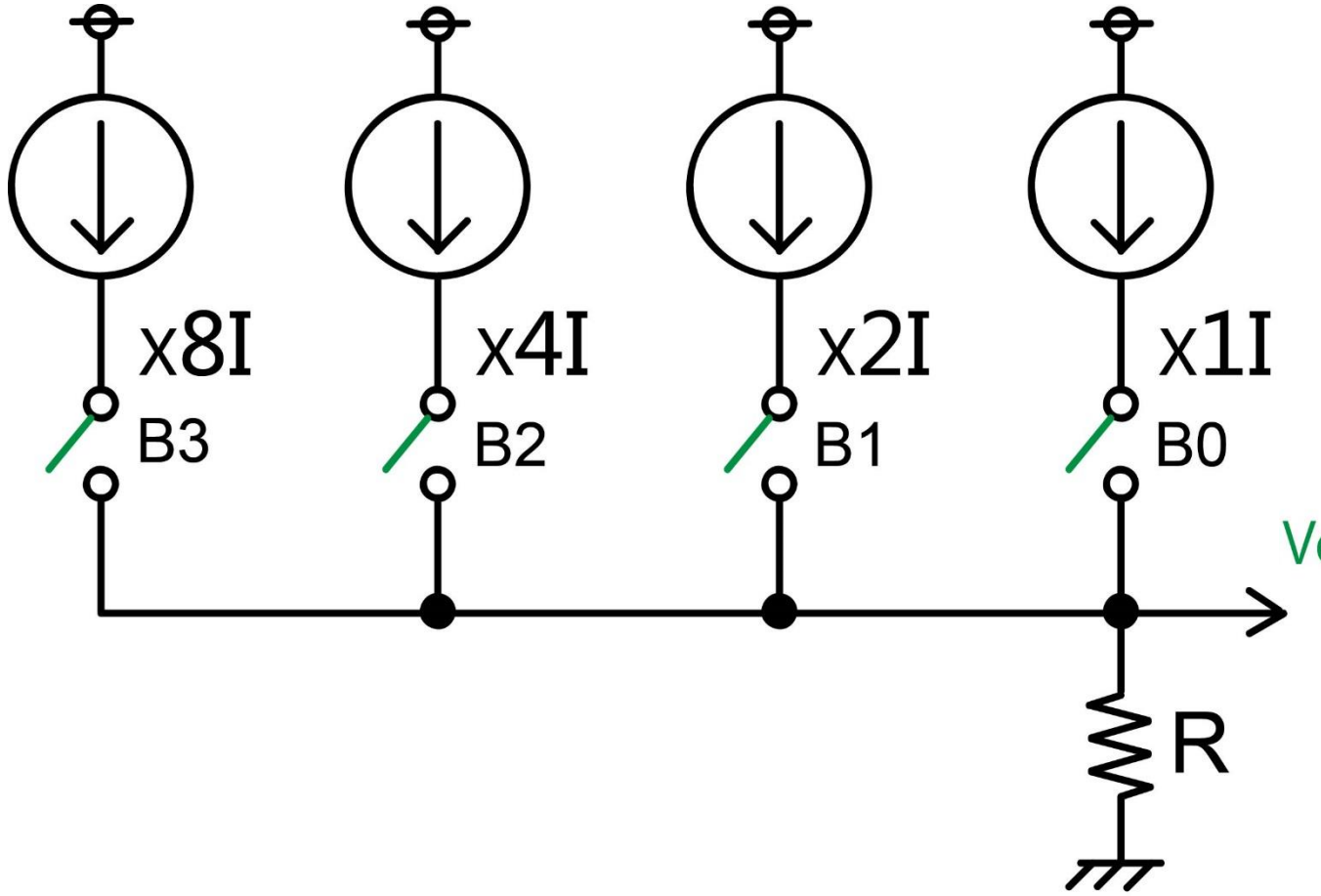


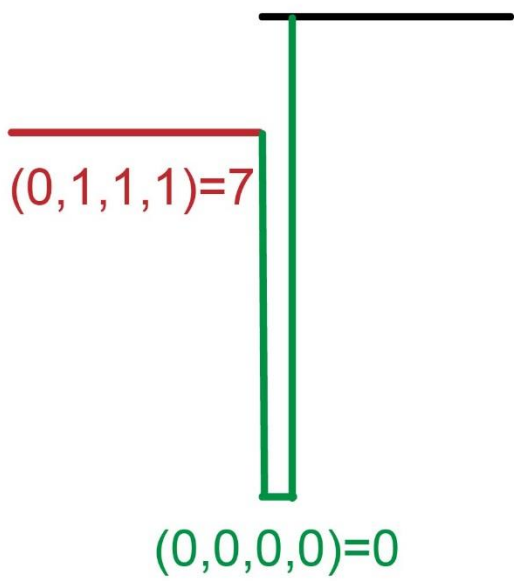

A big upward spike arises

When B3 switches first

# Generation of Glitch at Switching time

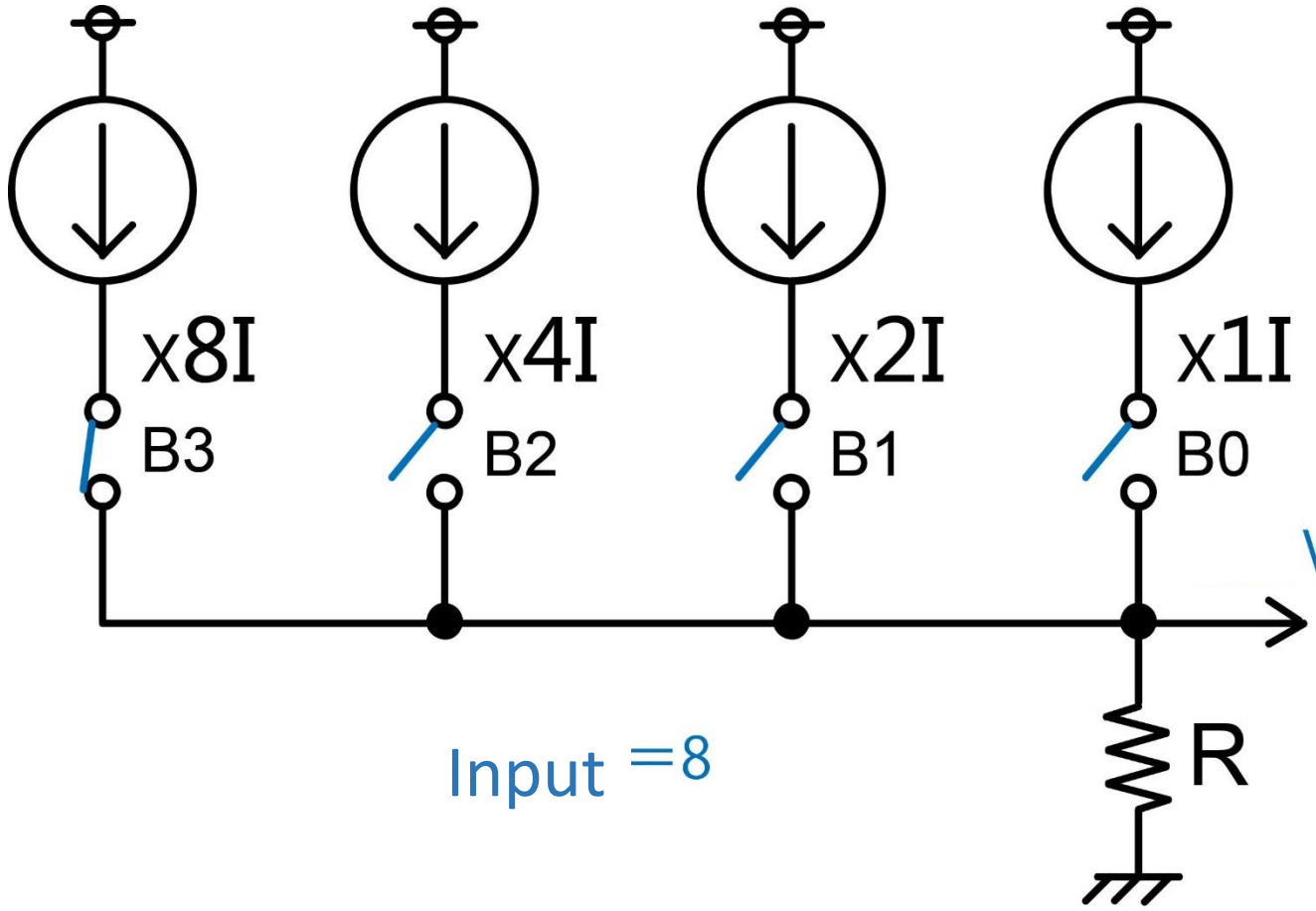


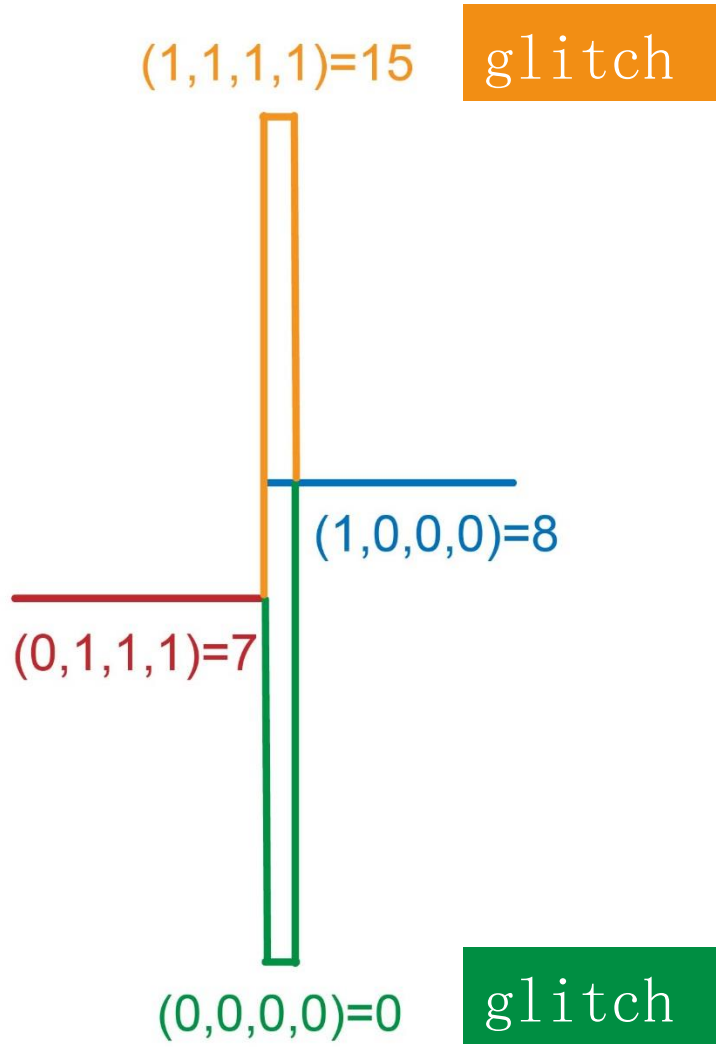

A big downward spike occur




When B3 switches last

# Generation of Glitch at Switching time



# Glitch Problem and Remedy

## Effects of Glitch

- Serious deterioration of images, videos, sounds

## Remedy

- Using high-order reconstruction filter
- Using track/hold circuitry at the DAC output

} Extra Space in IC, Expensive

- Using Gray-Code input DAC topologies

# OUTLINE

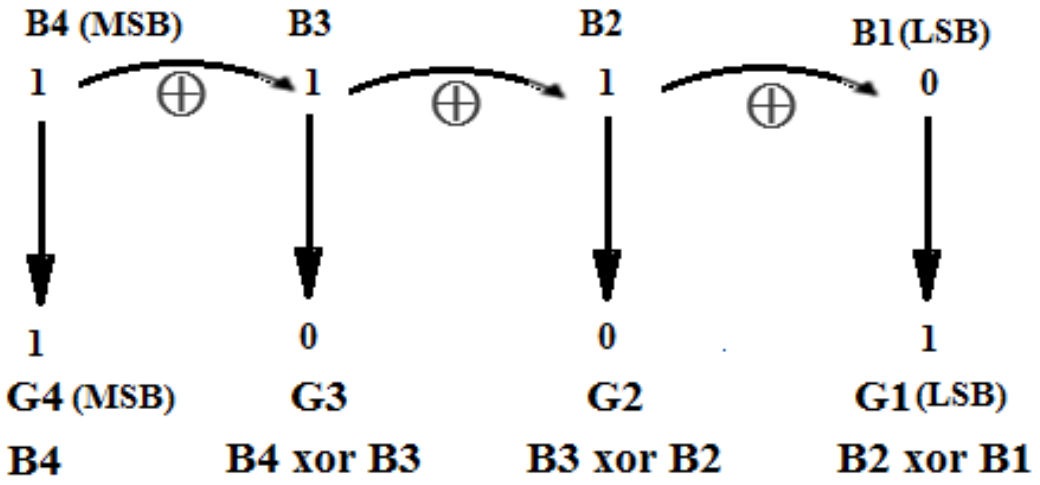
- Research Background・Objective
- What are Glitches
- Gray-code
- Gray-code Input DAC Architecture and Operation
- Simulation Verification by SPICE
- Conclusion

# Gray-Code

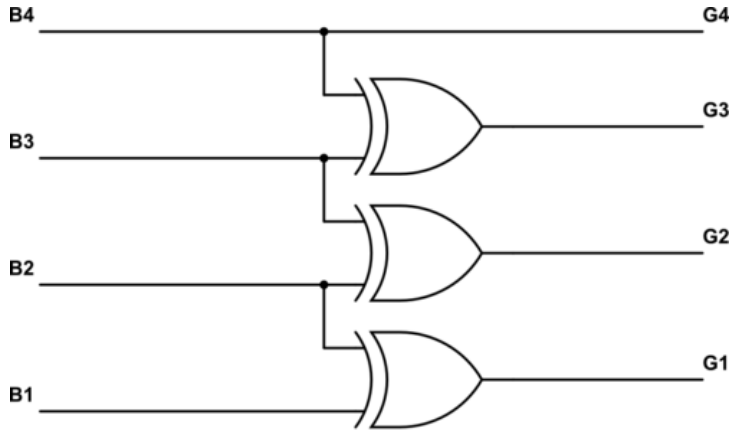
Gray-Code → Alternative representation of binary code

Two adjacent number → Only one bit change

$(G_n = B_{n+1} \oplus B_n)$




Binary to Gray code conversion diagram




Binary to Gray code converter

# Gray Code

Compare with Binary code and Gray code

| Decimal numbers | Binary Code | Gray Code |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

Binary code Multiple bits change at a time

Trigger more switches

Example. 1 → 2 --- 0001 → 0010 2 bits change

7 → 8--- 0111 →1000 all 4 bits change

Gray code Only one bit changes at a time

Triggers one switch

Example. 1 → 2 --- 0001 → 0011 one bit change

7 → 8 --- 0100→1100 one bit change

**Less glitches**

# OUTLINE

- Research Background・Objective
- What are Glitches
- Gray-code
- **Gray-code Input DAC Architecture and Operation**
- Simulation Verification by SPICE
- Conclusion

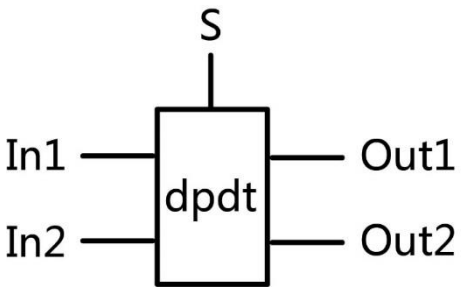
# Gray-code Input DAC Architecture and Operation

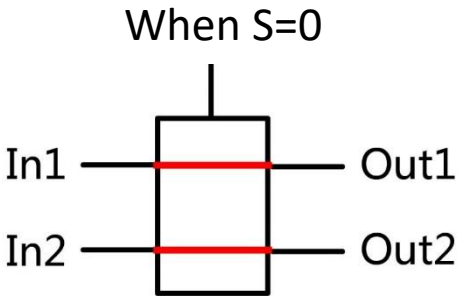
1. Current-steering Gray-Code DAC

2. Charge-mode Gray-Code DAC

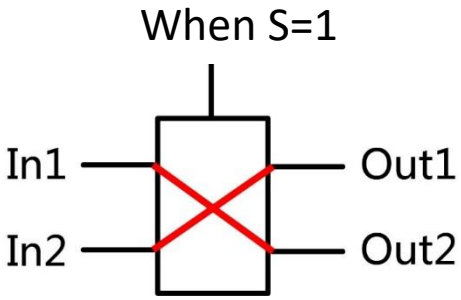
3. Voltage-mode Gray-Code DAC

# Current/Voltage Switch Matrix

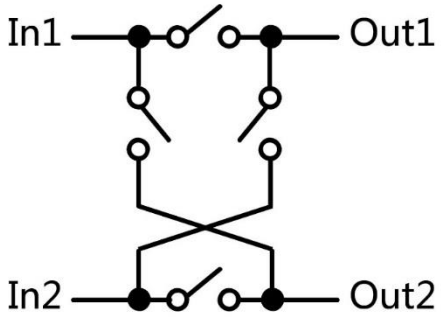


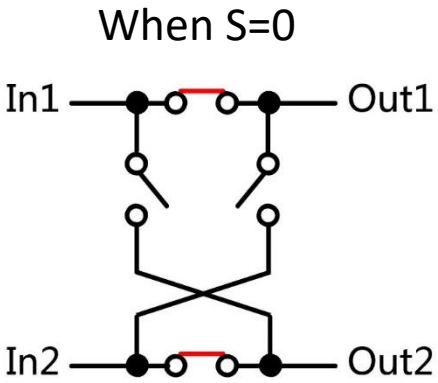



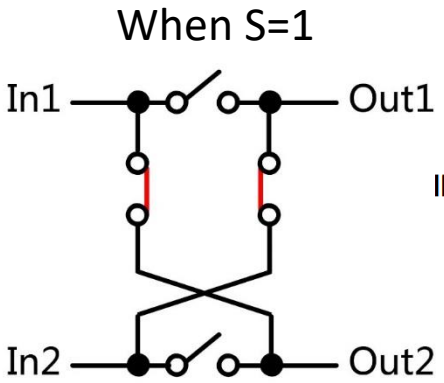

Parallel connection

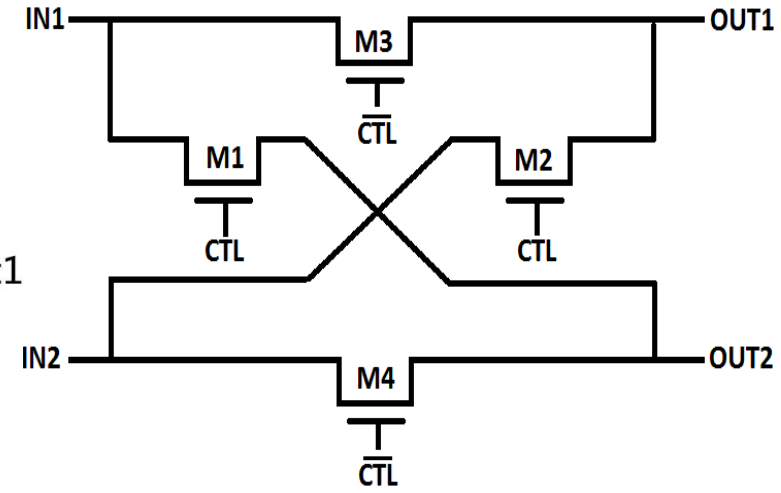



Cross connection



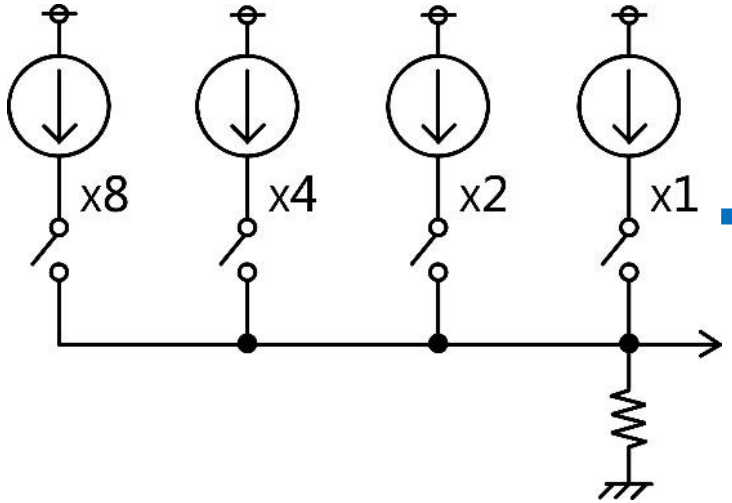



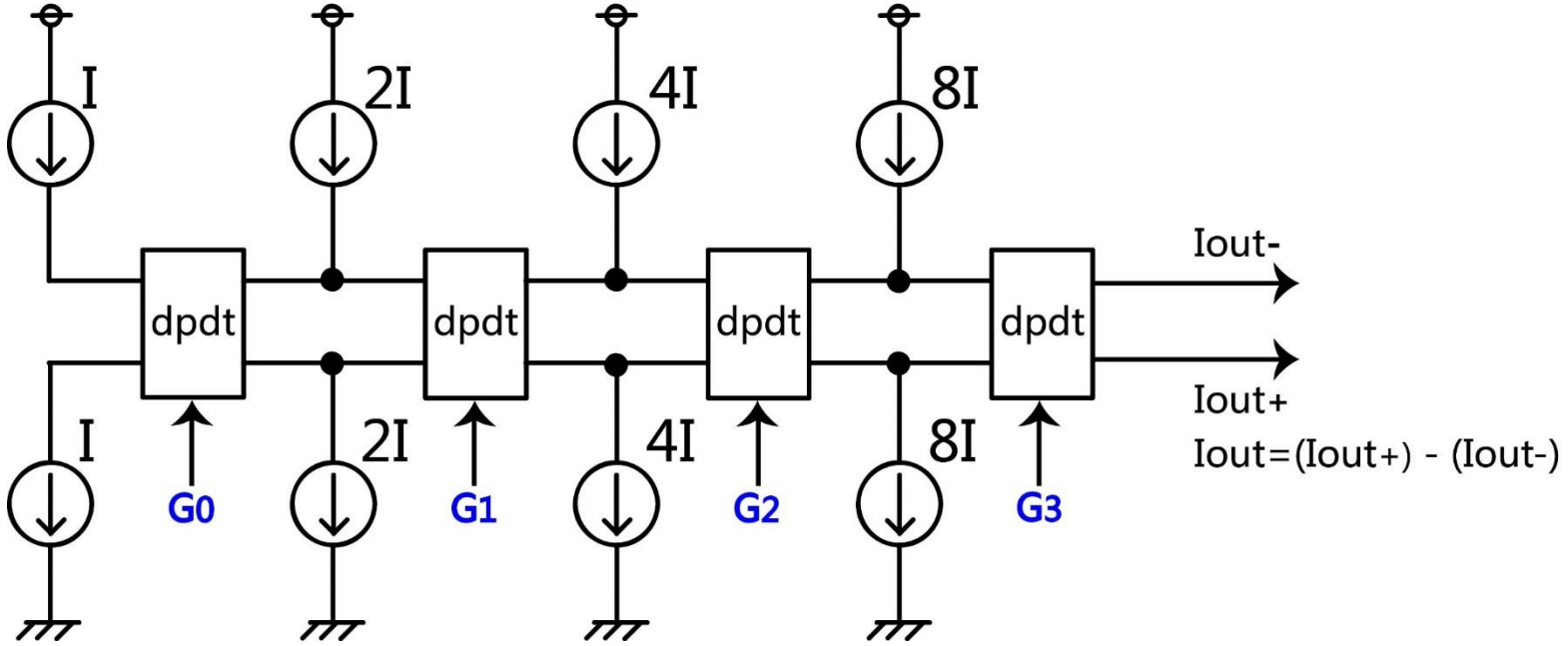




Switch is DPDT (Double-Pole Double-Throw)

# 1.Current-steering Gray-Code DAC

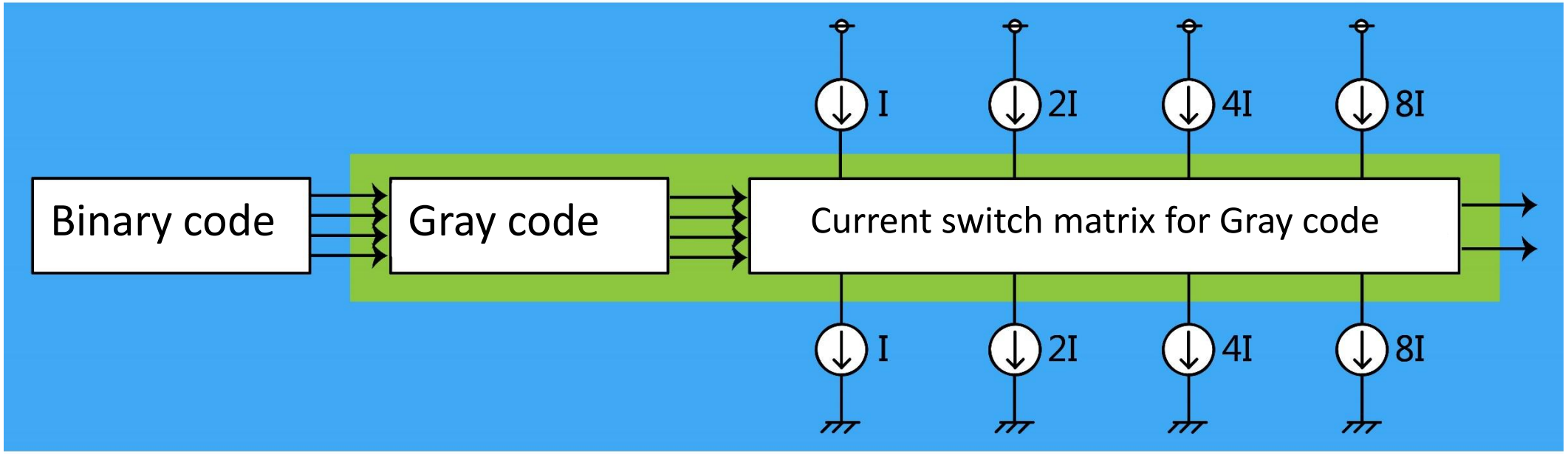



Conventional Binary-Weighted current-steering DAC




Gray-Code input current-steering DAC

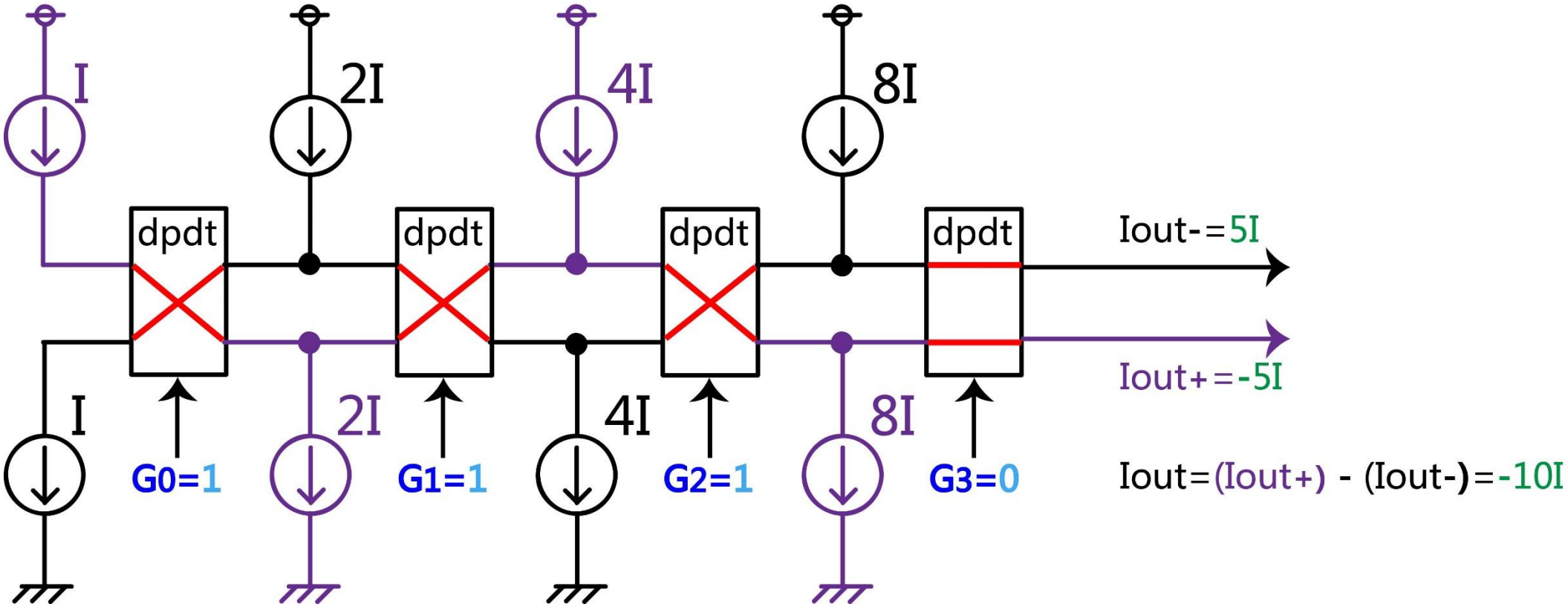
# Code Conversion




Code domain in Gray-code input current-steering DAC

# A Gray-code input current-steering DAC (data=5)

Data=5

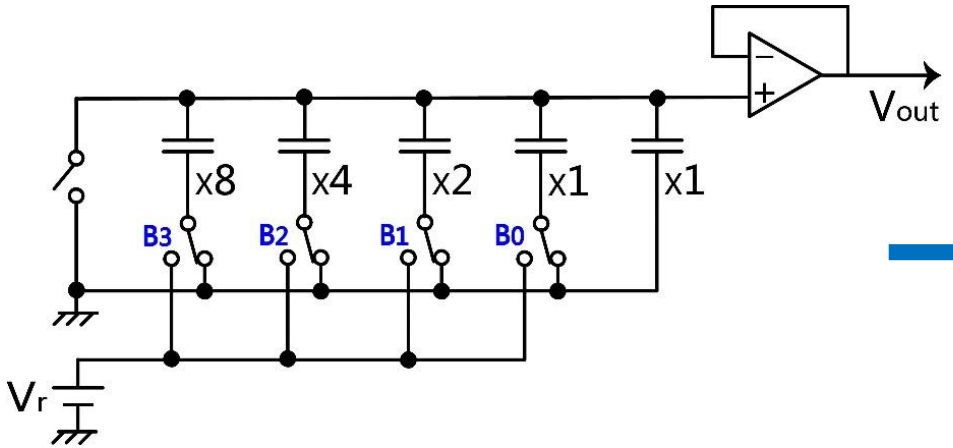



$$I_{out-} = -I + 2I - 4I + 8I = 5I$$

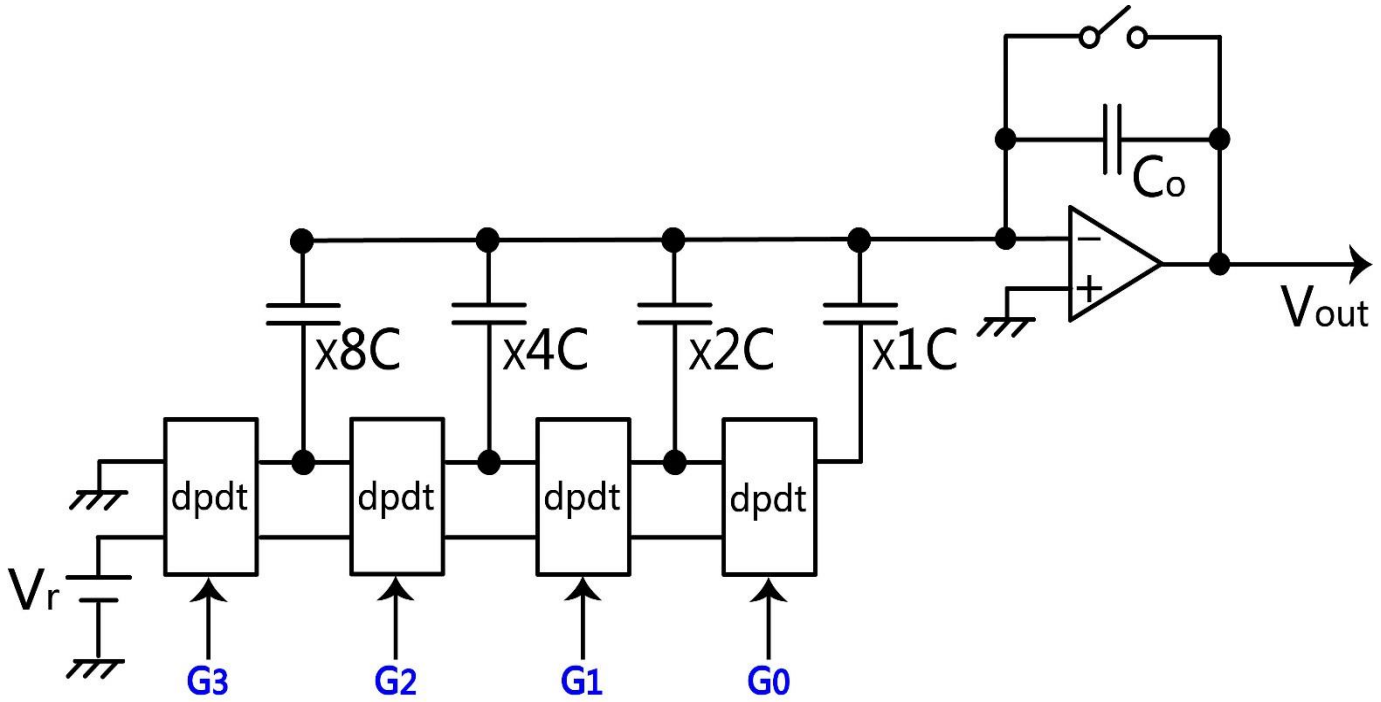
$$I_{out+} = I - 2I + 4I - 8I = -5I$$

$$I_{out} = (I_{out+}) - (I_{out-}) = -10I$$

# 2.Charge-mode Gray-code DAC

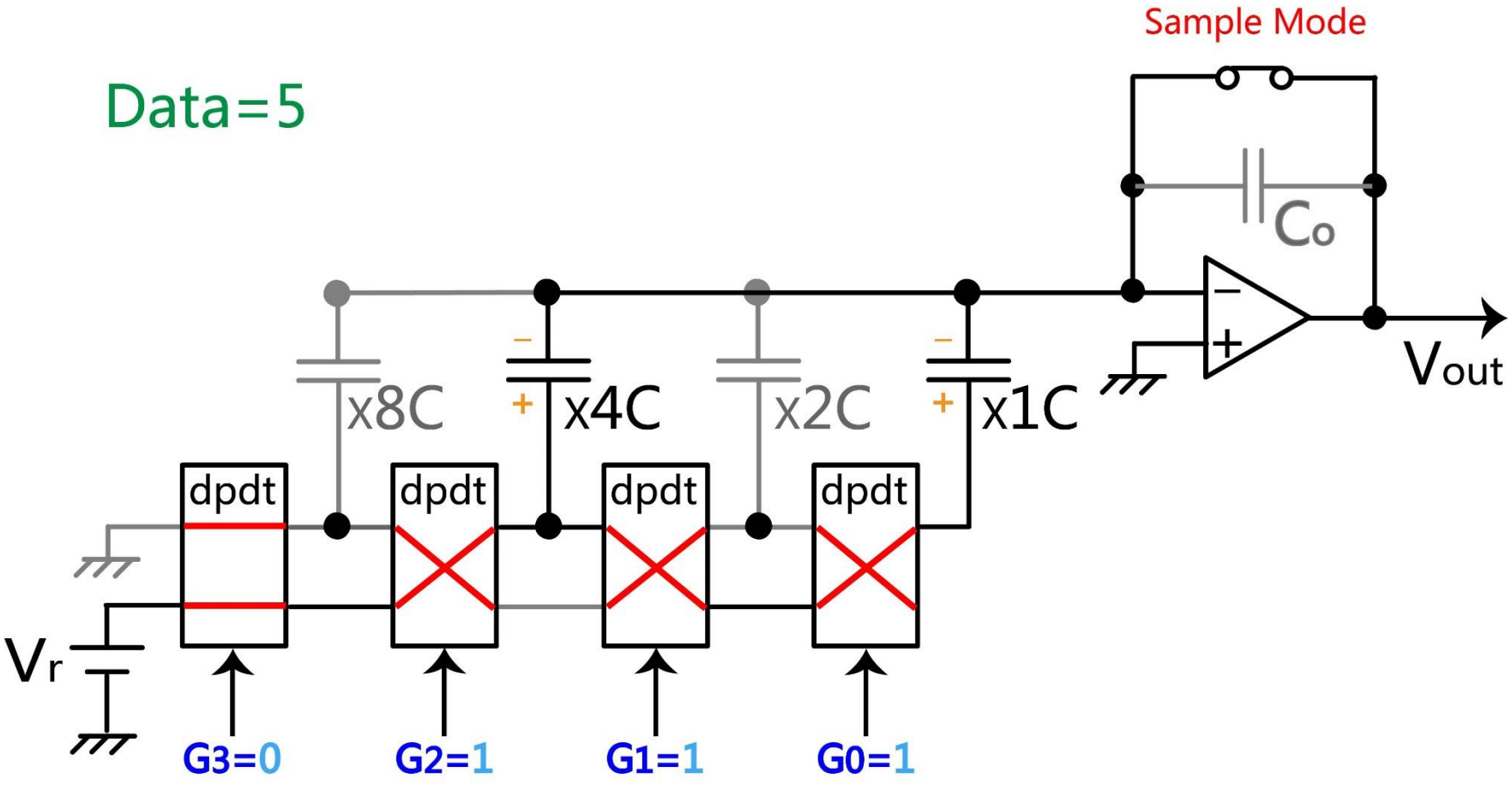



A binary-weighted capacitor DAC

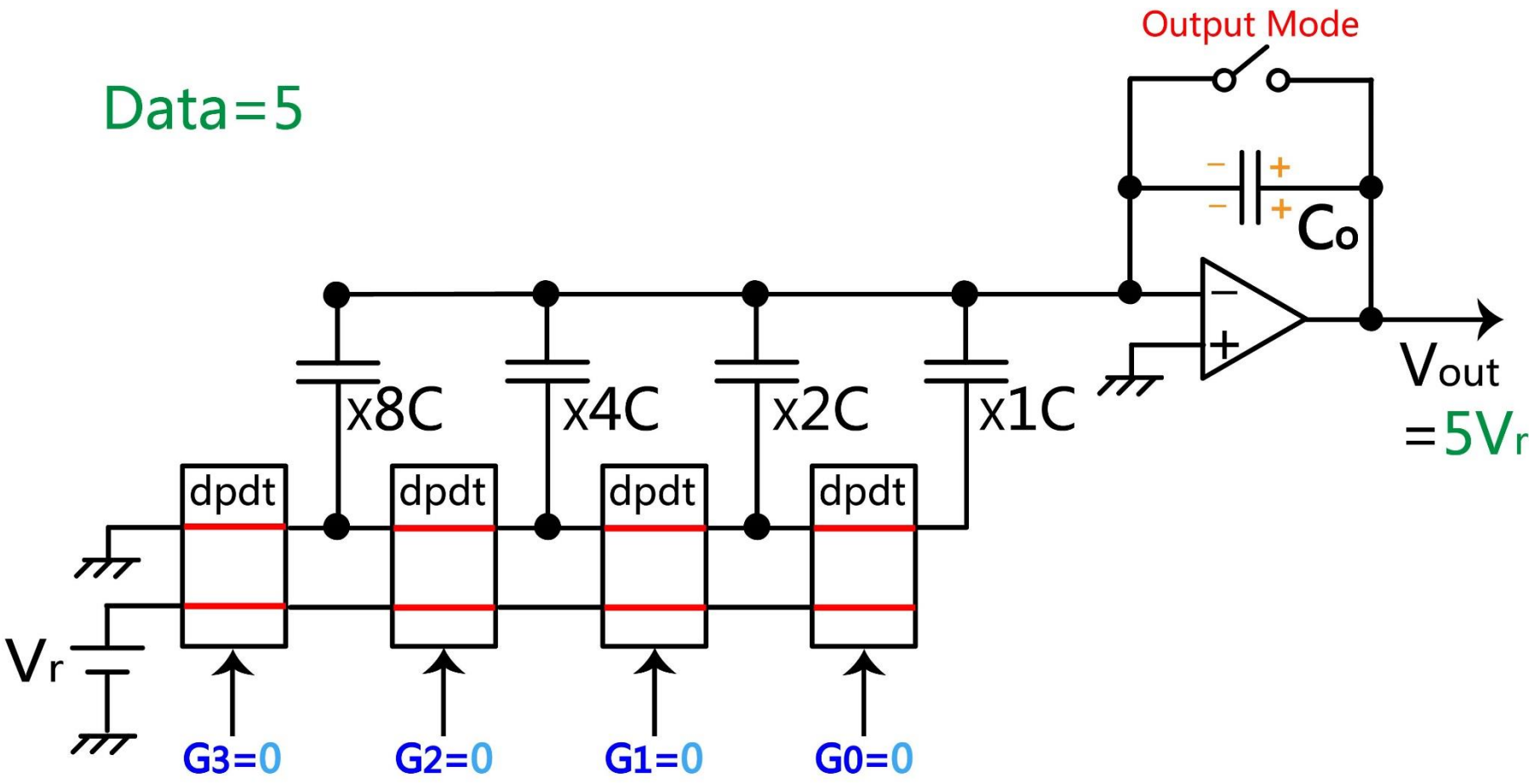



A Gray-code input charge-mode DAC

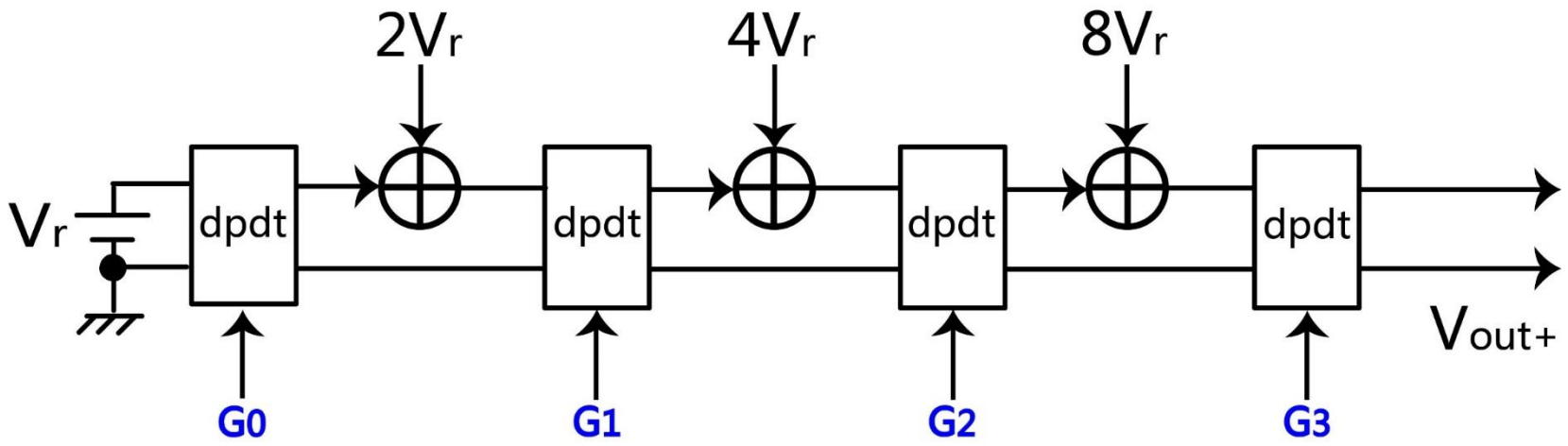
# Sample Mode of a Gray-Code Input Charge-Mode DAC (data=5)



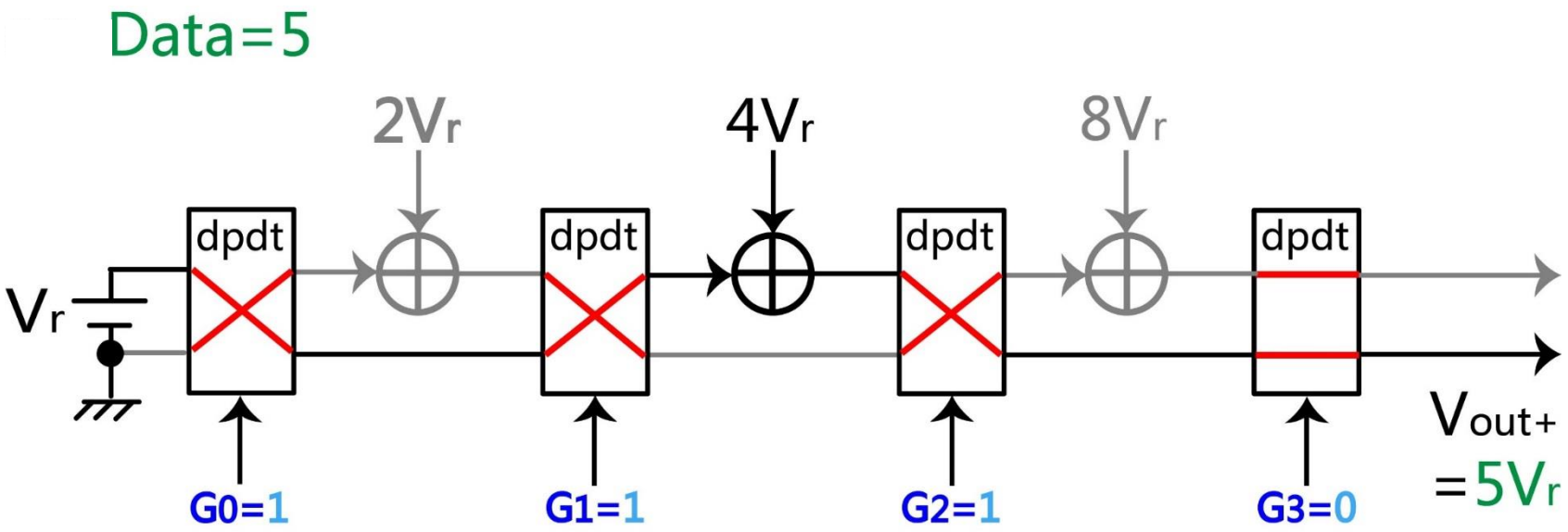
# Sample Mode of a Gray-Code Input Charge-Mode DAC (data=5)



# 3.Voltage-mode Gray-Code DAC



# A Gray-Code Input Voltage-mode DAC (data=5)




$$V_{out+} = V_r + 4V_r = 5V_r$$

# OUTLINE

- Research Background・Objective
- What are Glitches
- Gray-code
- Gray-code Input DAC Architecture and Operation
- Simulation Verification by SPICE
- Conclusion

# Simulation Verification by SPICE

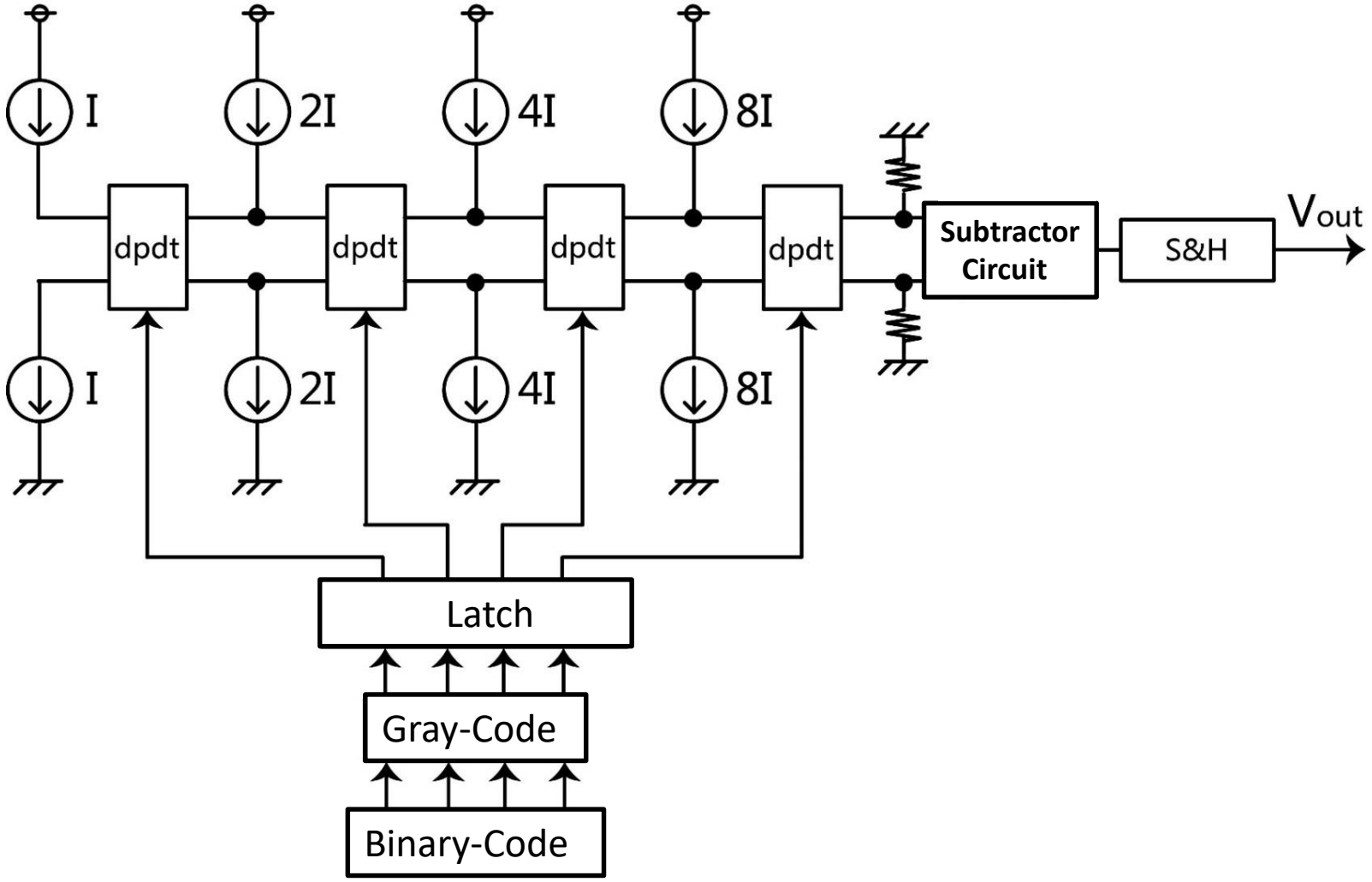
1.Simulation of current-steering Gray-Code DAC

2.Simulation of charge-mode Gray-Code DAC

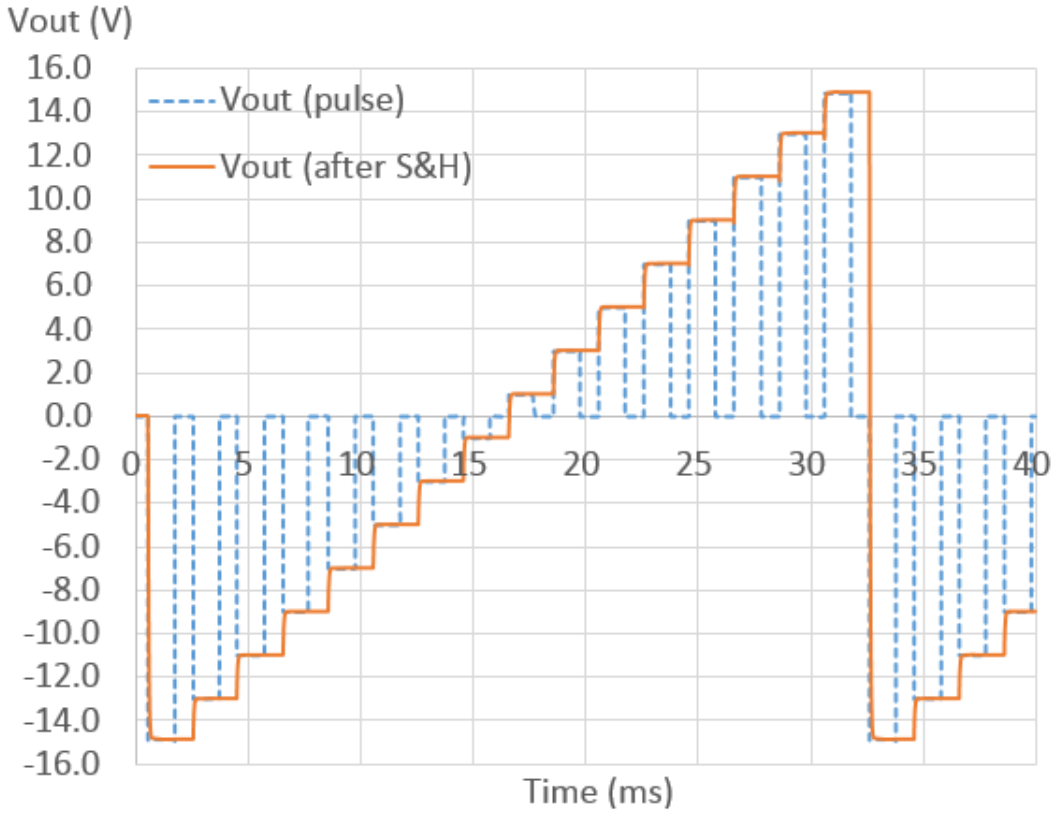
3.Simulation of voltage-mode Gray-Code DAC

4.Verification of glitch reduction

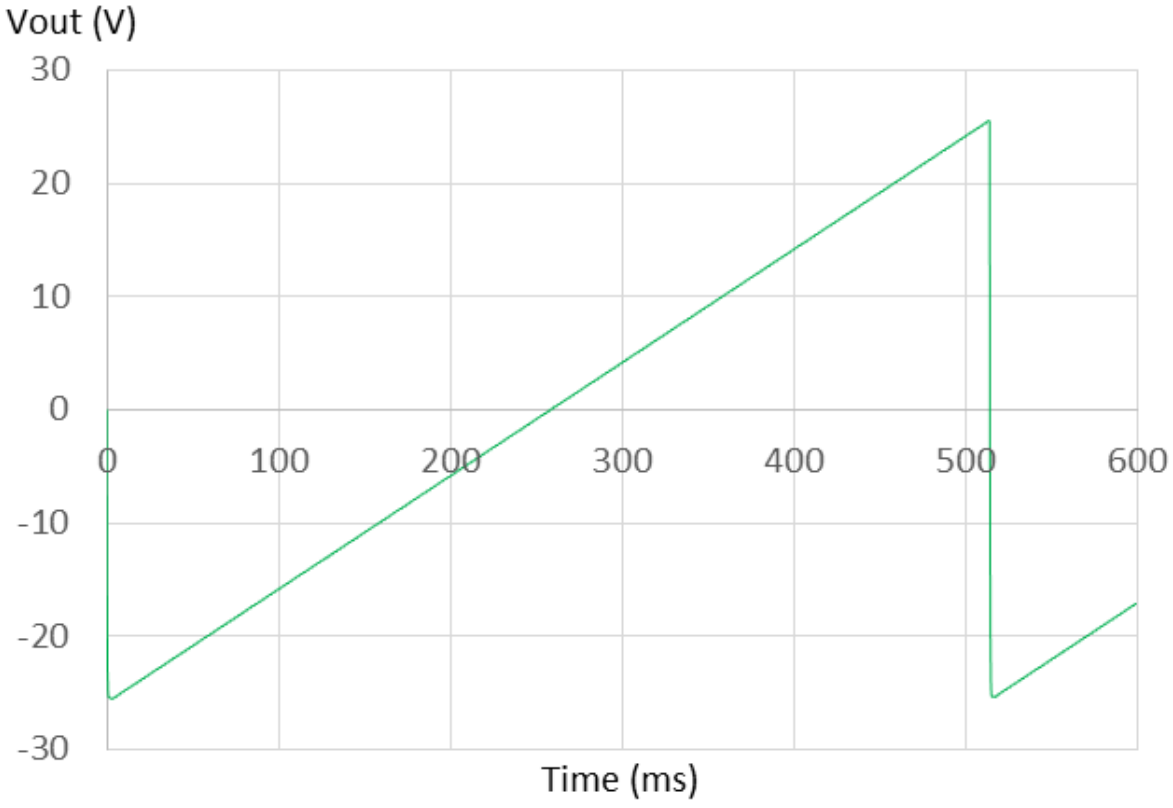
# 1.SPICE Realization of Current-Steering of Gray-Code Input



# 1.Simulation of current-steering Gray-Code DAC

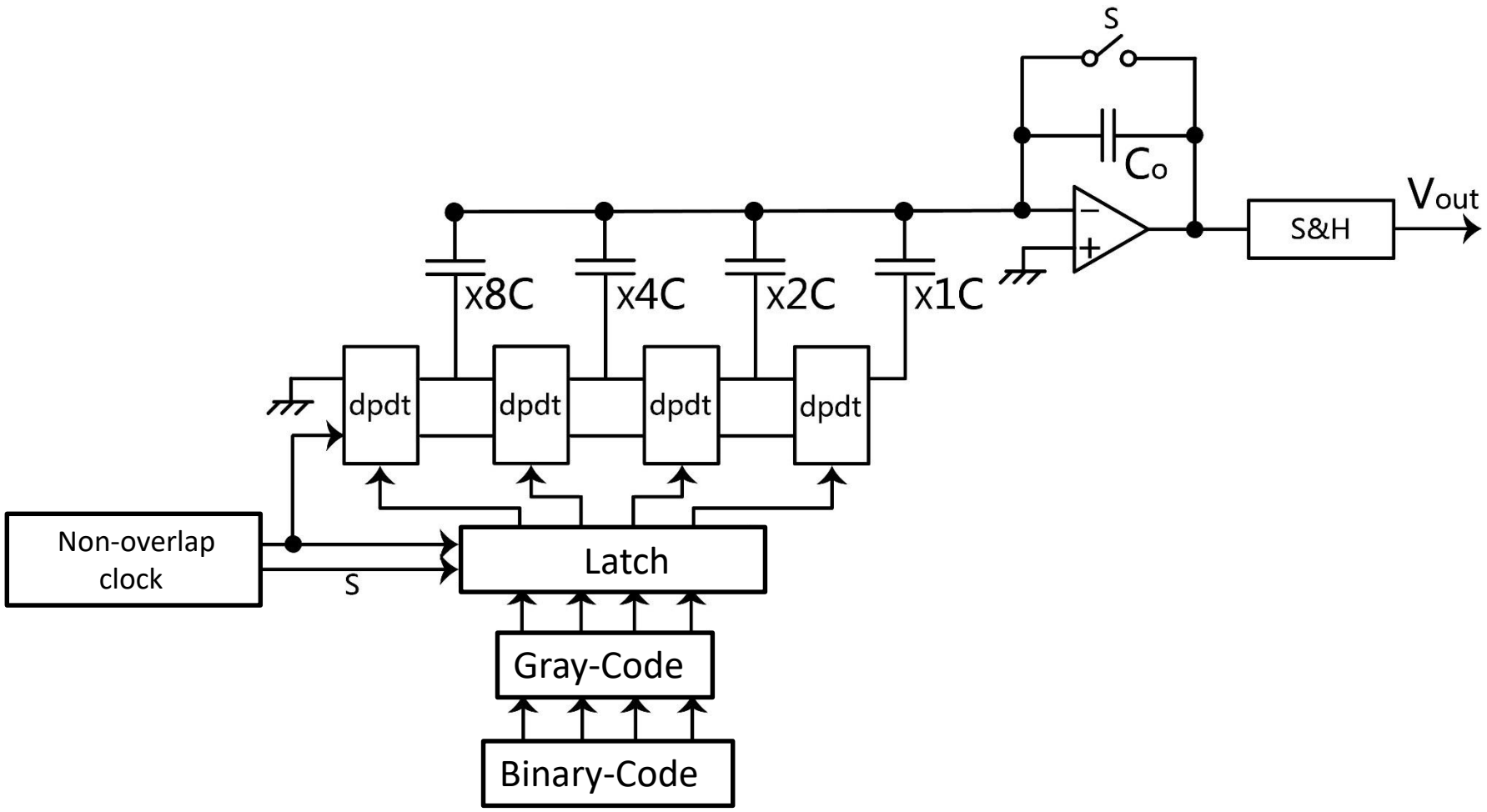



4bit Current-steering DAC

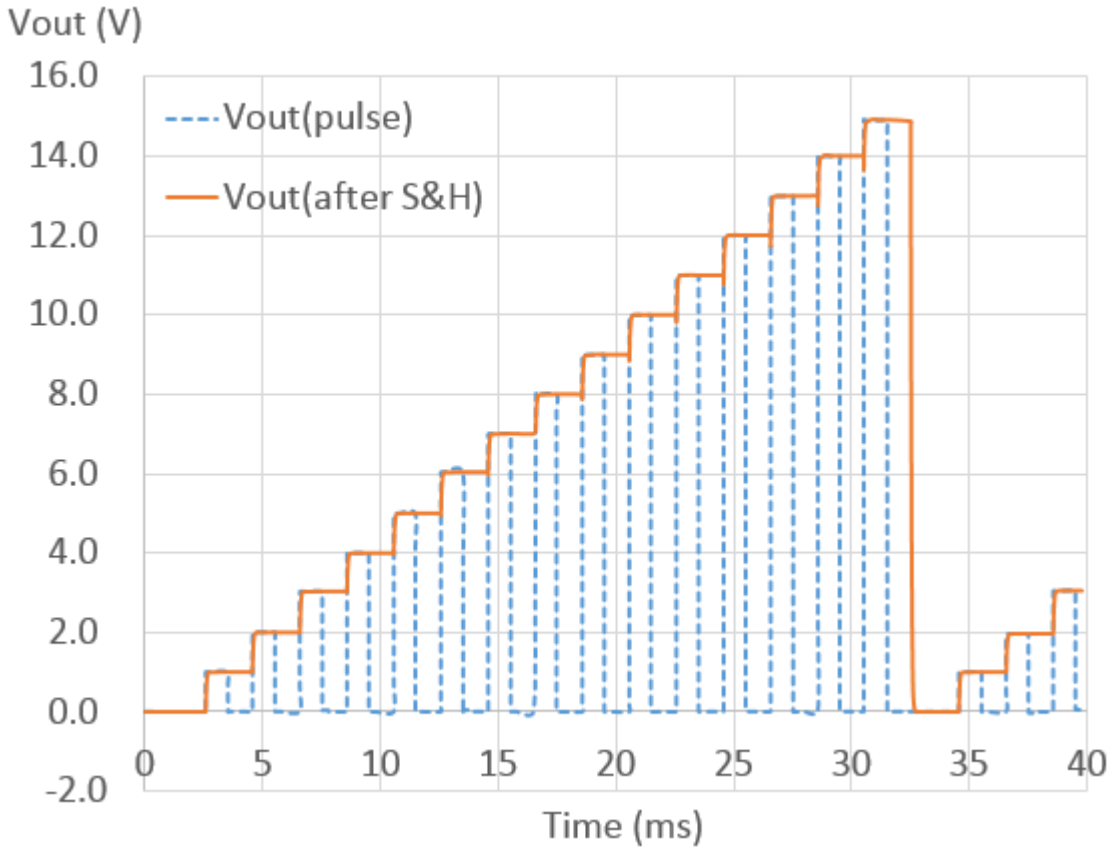



8bit Current-steering DAC

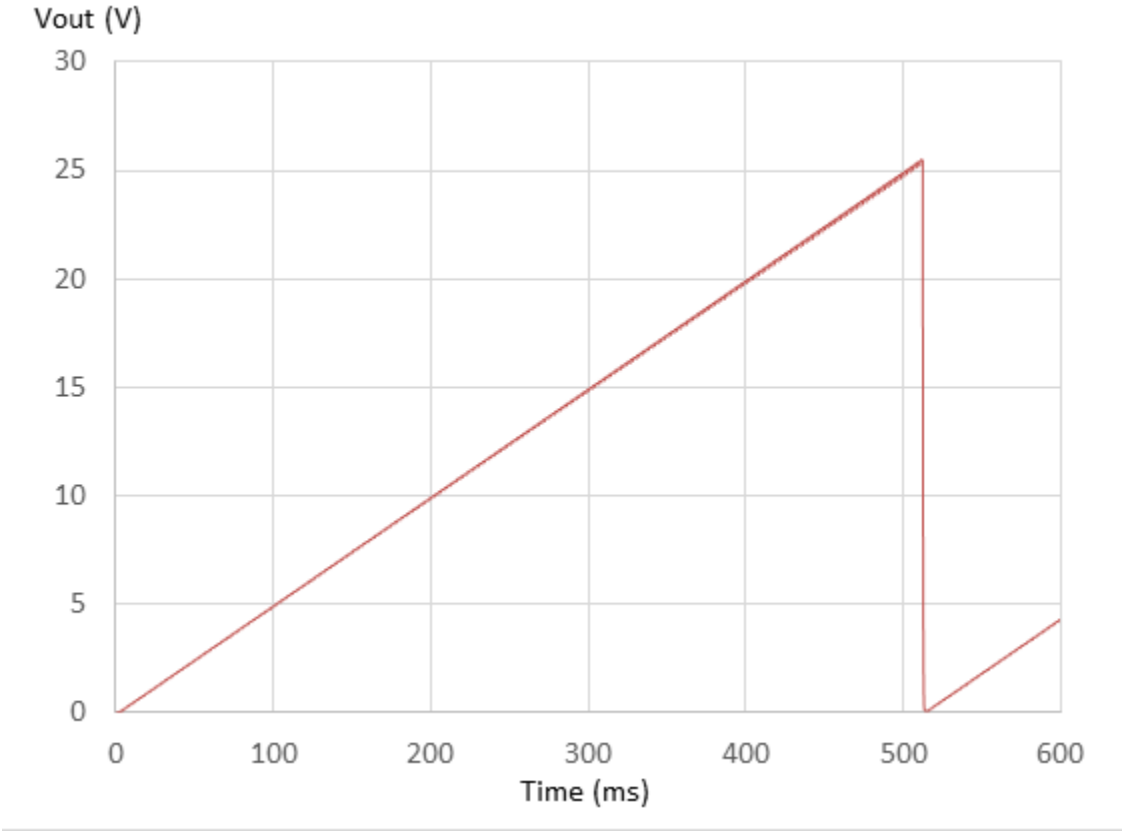
# 2.SPICE Realization of charge-mode of Gray-Code Input



# 2. Simulation of charge-mode Gray-Code DAC

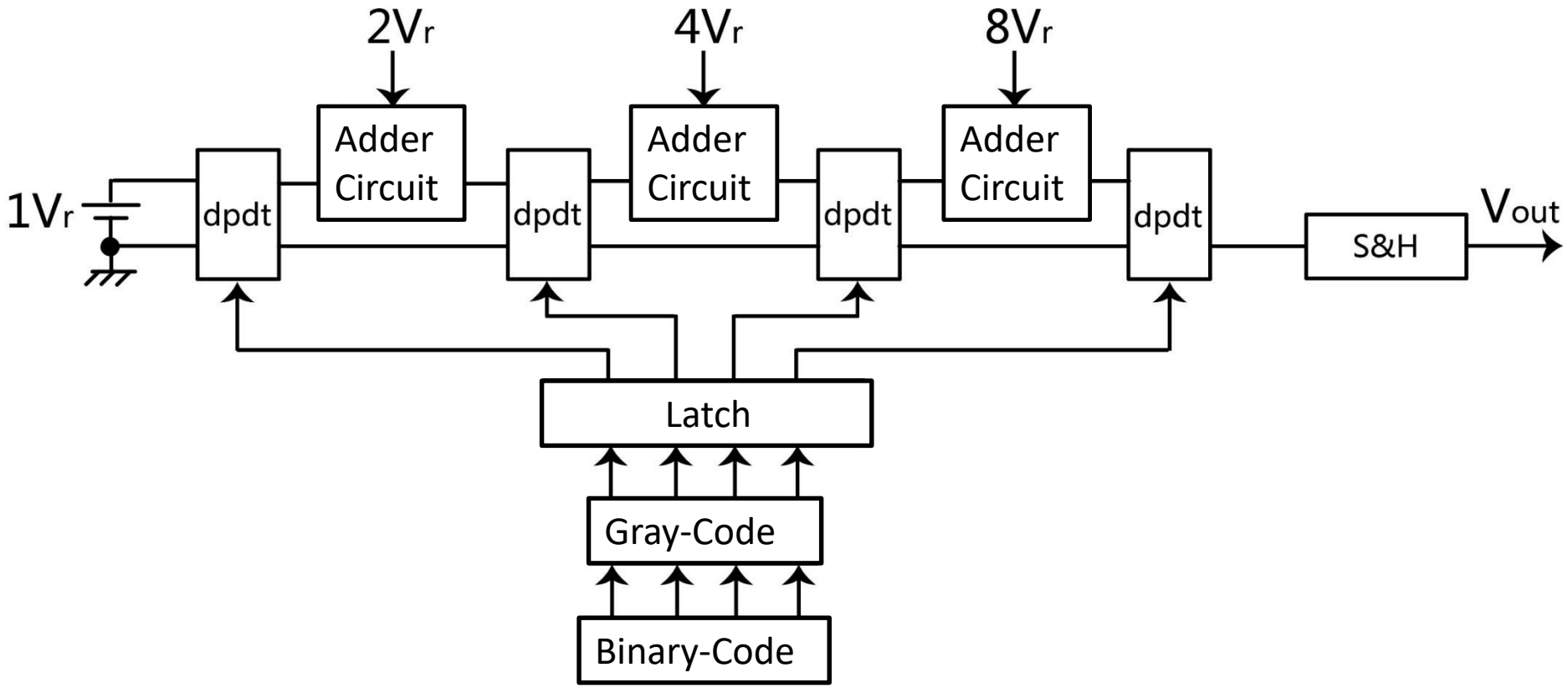



4bit Charge-mode DAC

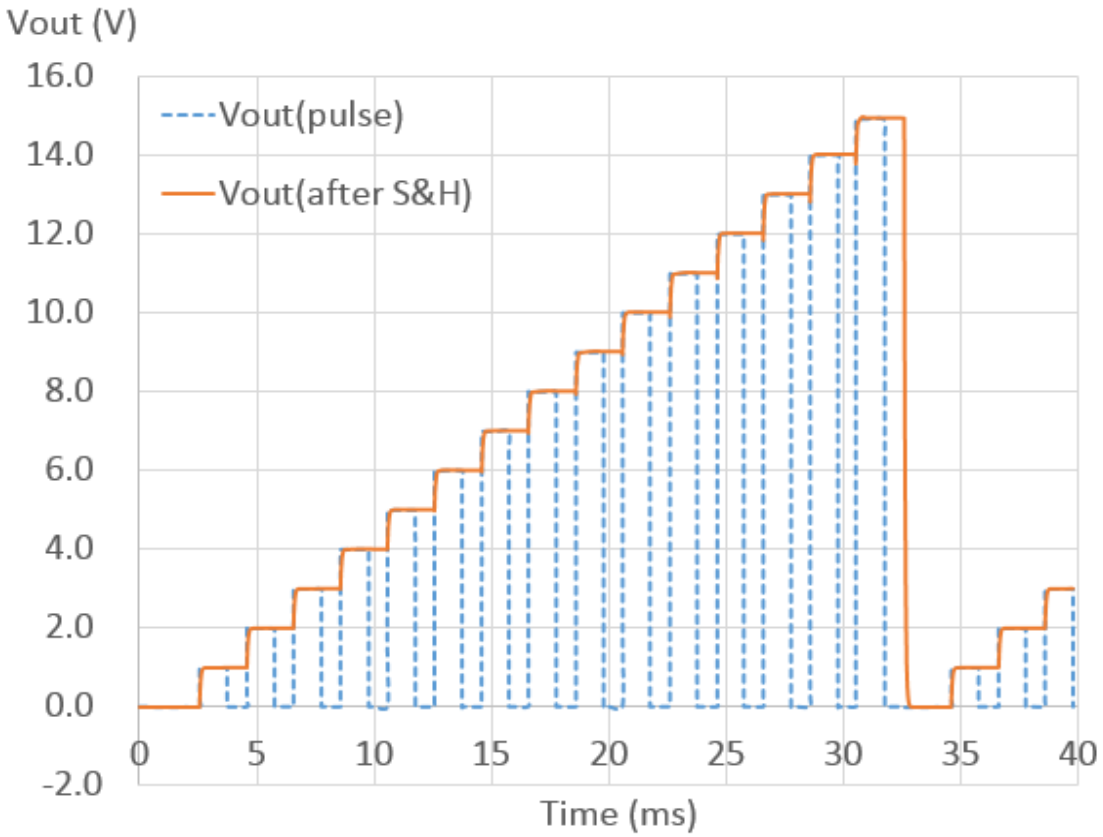



8bit Charge-mode DAC

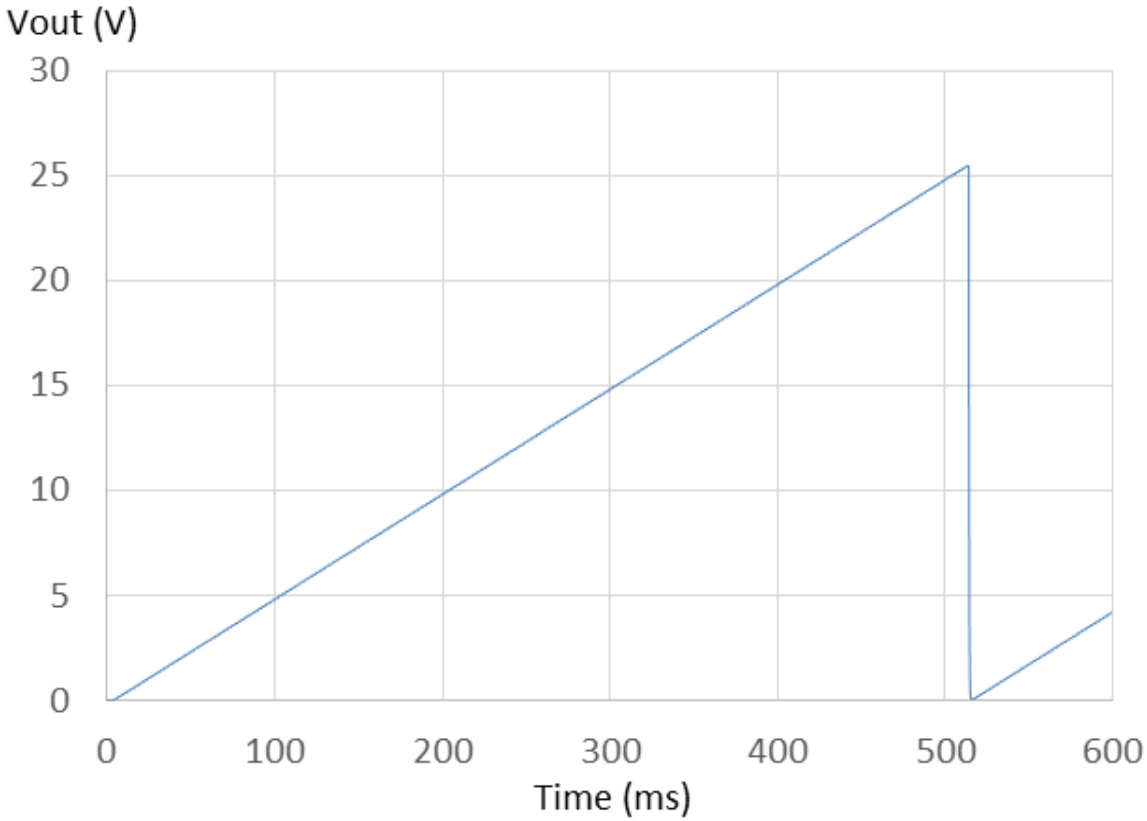
# 3.SPICE Realization of Voltage-mode of Gray-Code Input



# 3. Simulation of Voltage-mode Gray-Code DAC

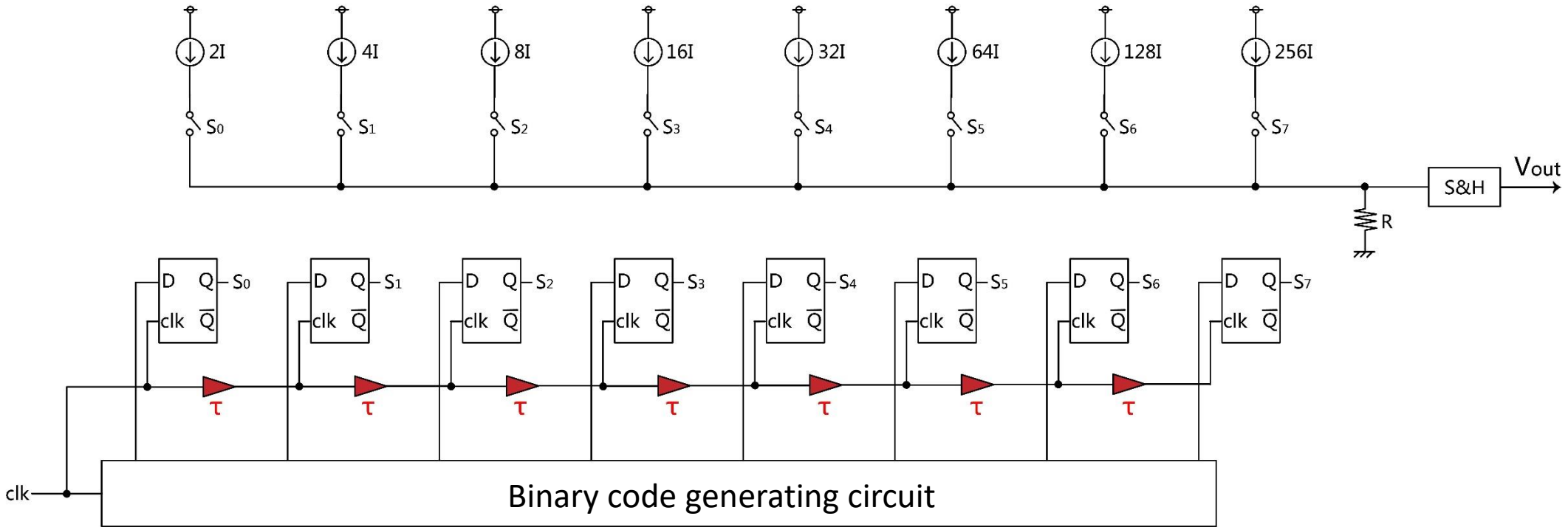



4bit Voltage-mode DAC



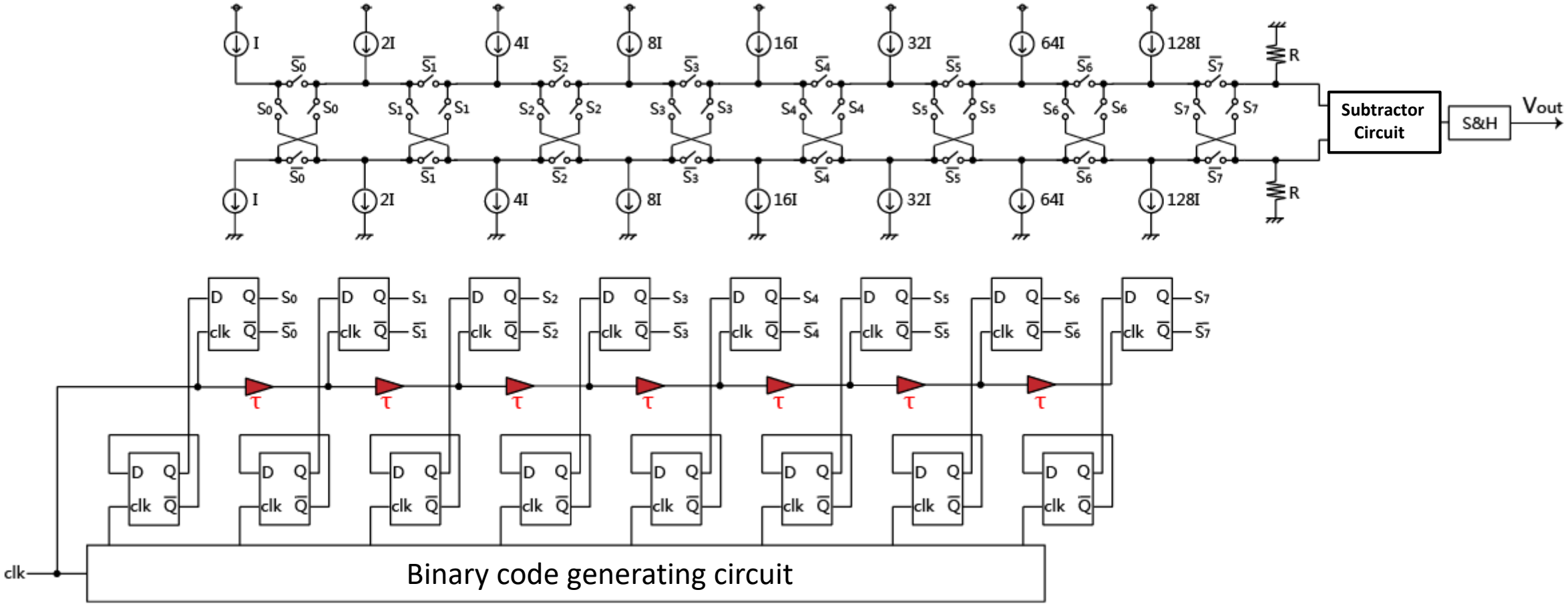

8bit Voltage-mode DAC

# 4.Verification of glitch reduction



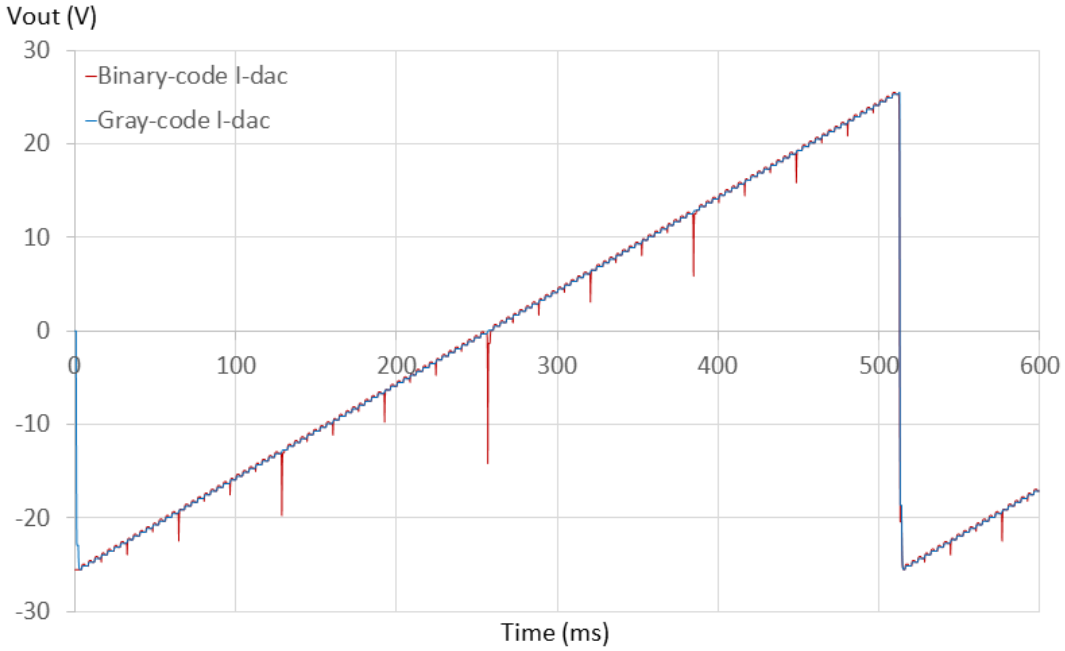

Conventional Current-Steering DAC with switching delay (8bit)

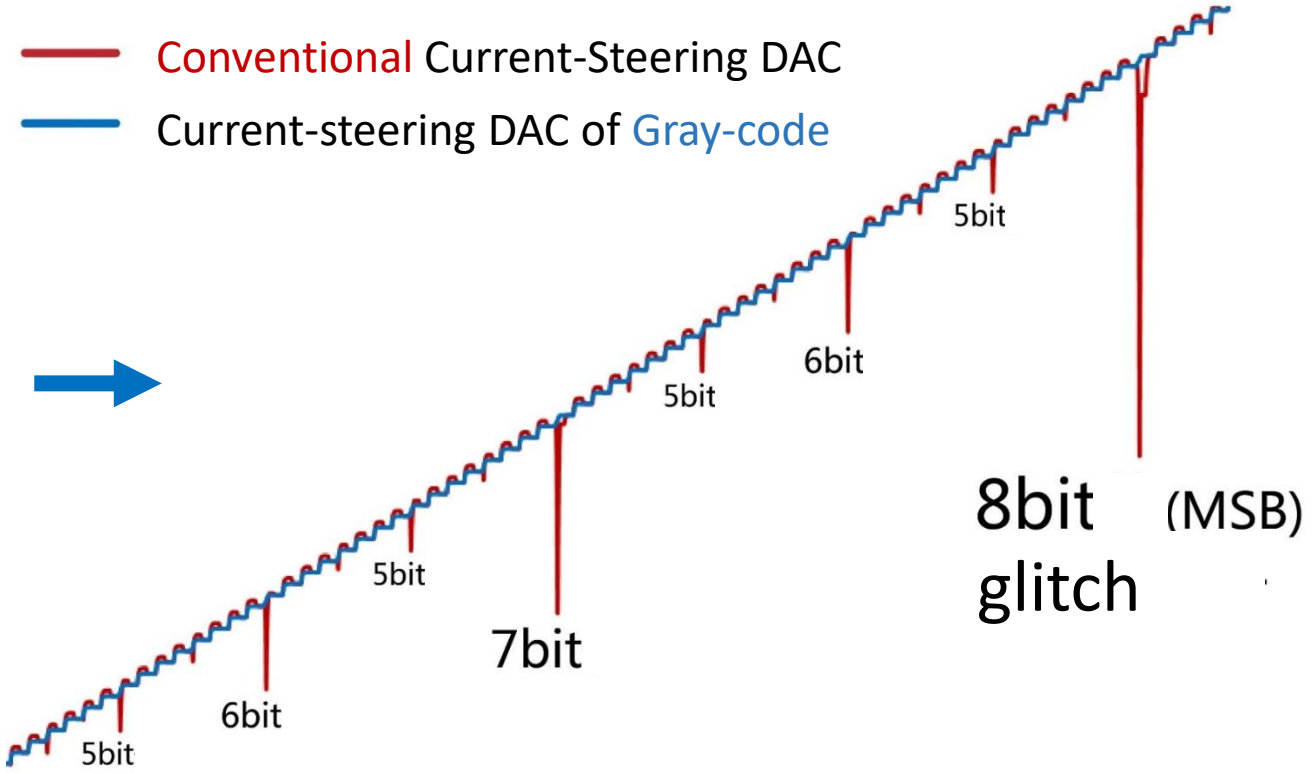
# 4.Verification of glitch reduction




Current-Steering Gray-code input DAC with switching delay (8bit)

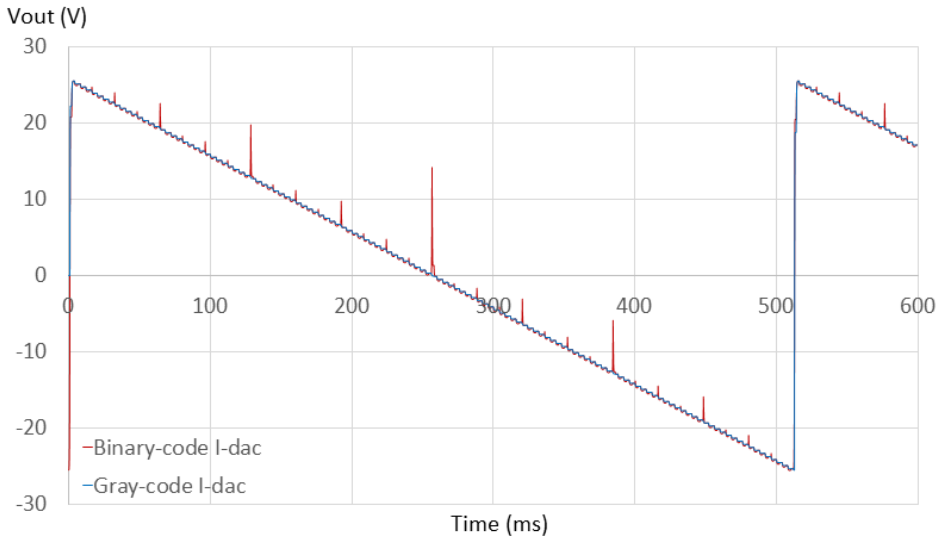
# 4.Simulation Result (Up Sweeping)

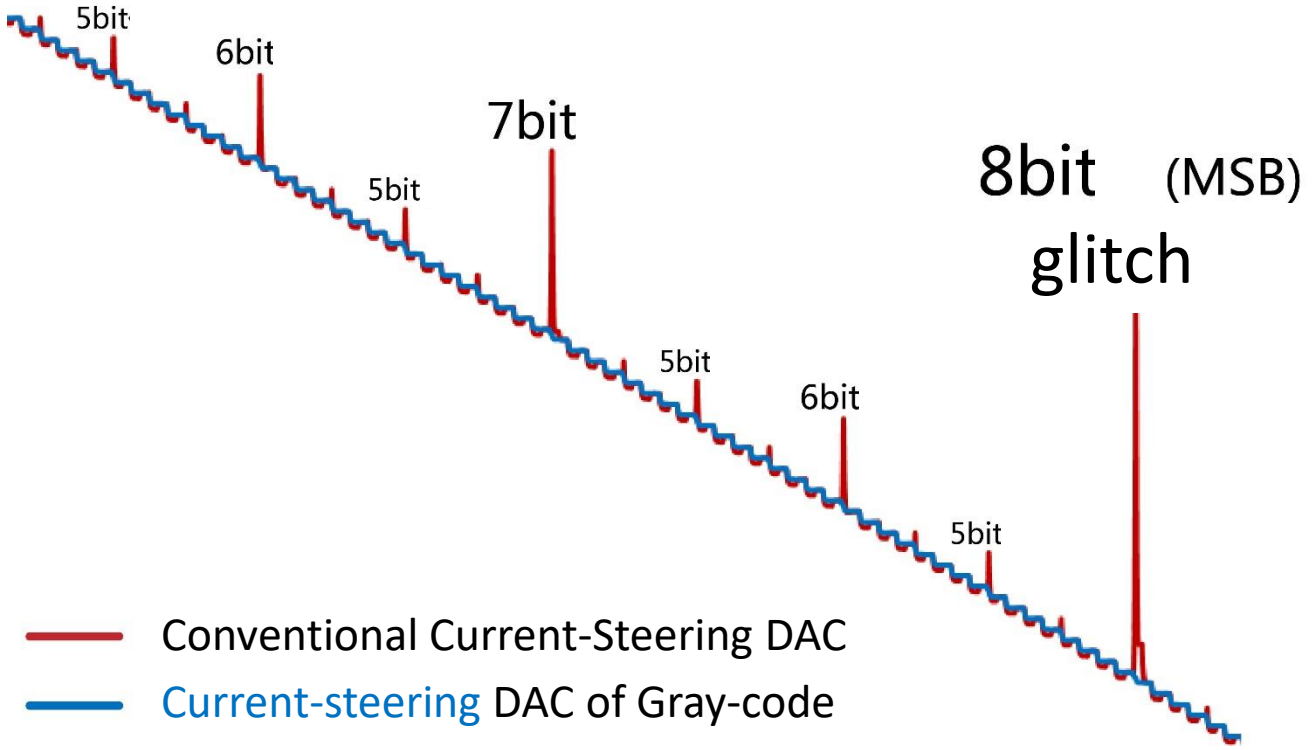



Conventional Current-Steering DAC vs. Current-steering DAC of Gray-code

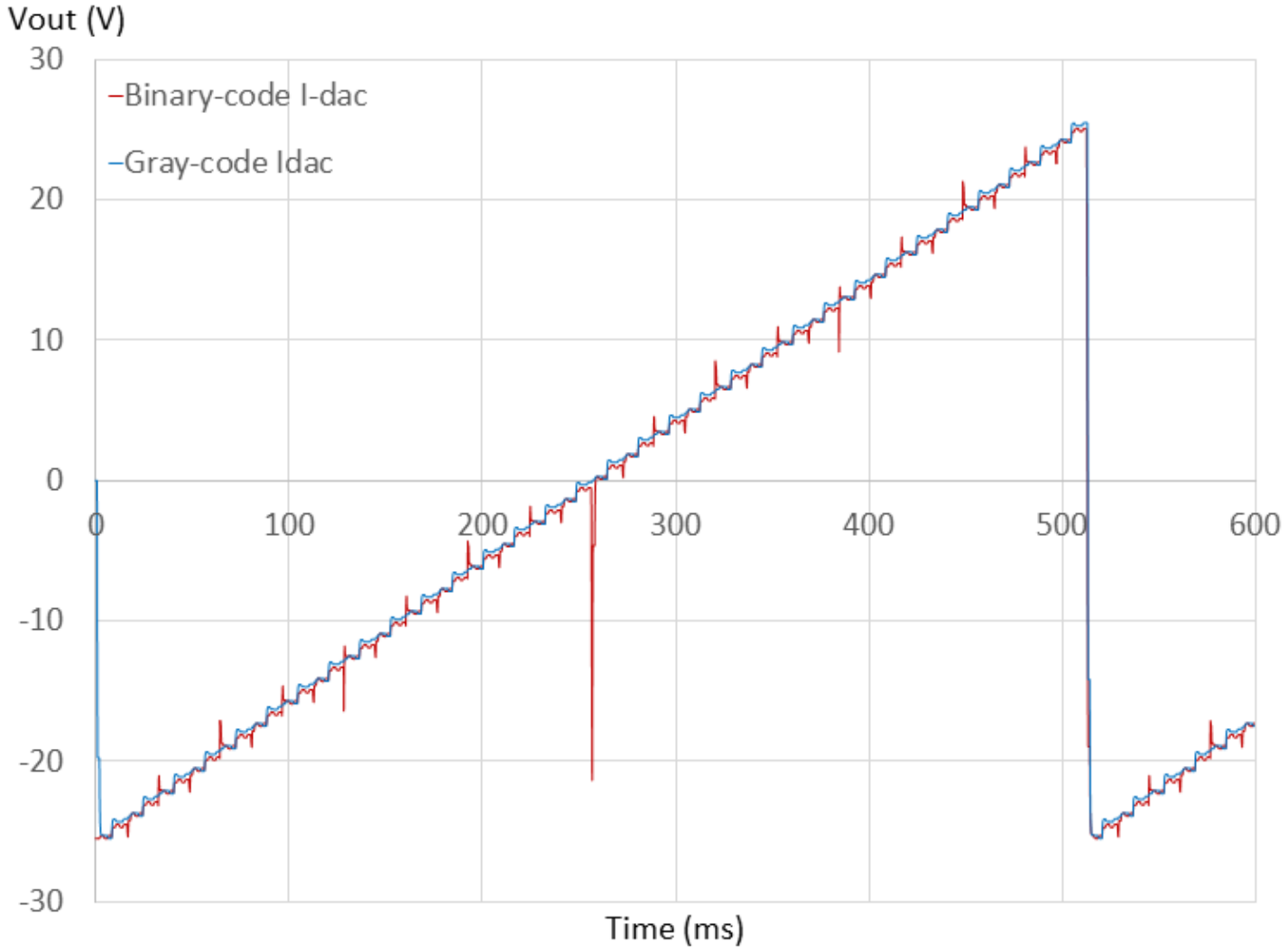
# 4.Simulation Result (Down Sweeping)




Conventional Current-Steering DAC vs. Current-steering DAC of Gray-code

# 4.Simulation Result (Random Switching Delay)

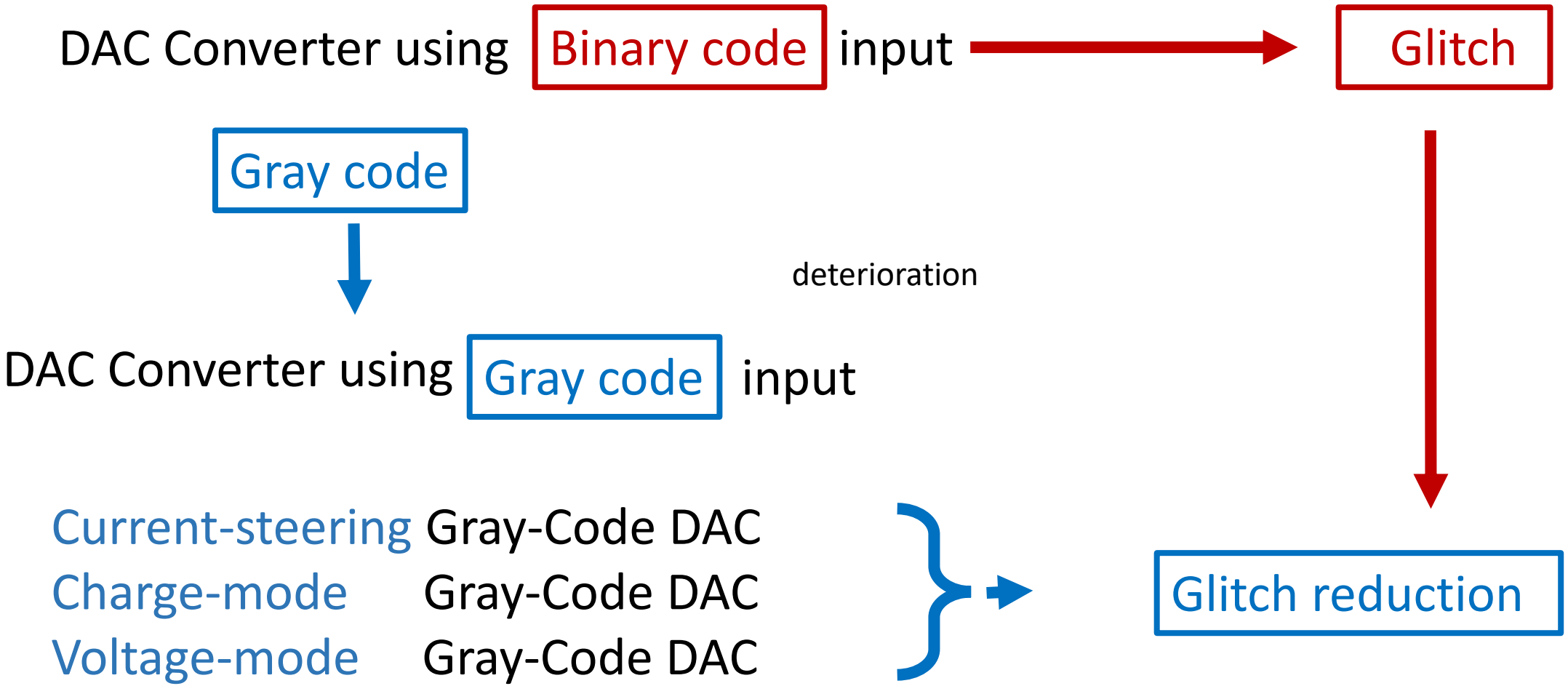



Conventional Current-Steering DAC vs. Current-steering DAC of Gray-code

# OUTLINE

- Research Background・Objective
- What are Glitches
- Gray-code
- Gray-code Input DAC Architecture and Operation
- Simulation Verification by SPICE
- Conclusion

# Conclusion

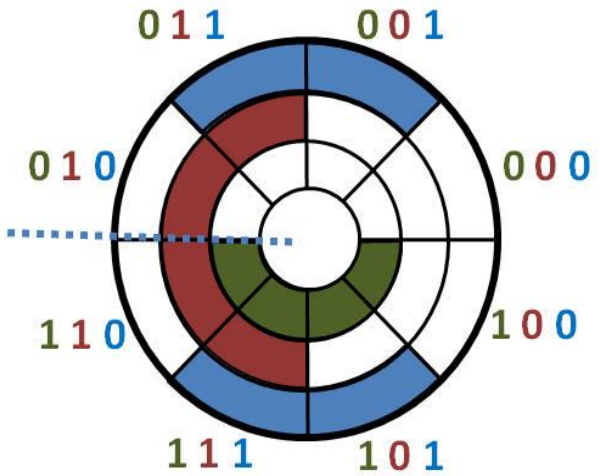

DAC Converter using
Binary code
input
Glitch
Gray code
deterioration
DAC Converter using
Gray code
input
Current-steering Gray-Code DAC
Charge-mode Gray-Code DAC
Voltage-mode Gray-Code DAC
Glitch reduction

# Final statement

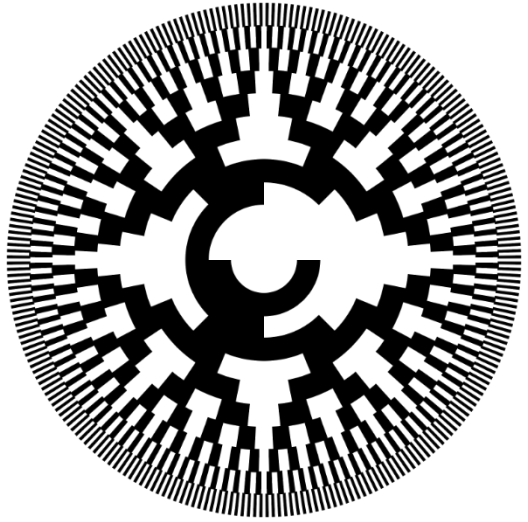
- Coding method can lead to robust mixed-signal circuit design.

  Gray code was invented by Frank Gray at Bell Lab in 1947.


FRANK GRAY and A. L. Johnsrud in television booth. Behind the glass panels at sides and top are the photo-electric cells.




| $Q_2Q_1Q_0$ |
|---|
| 000 |
| 001 |
| 011 |
| 010 |
| 110 |
| 111 |
| 101 |
| 100 |



# Thank you for listening

谢谢

# Study of Gray Code Input DAC for Glitch Reduction


群馬大学
GUNMA UNIVERSITY

***Adhikari Gopal  Richen Jiang  Haruo Kobayashi**

**Kobayashi Laboratory, Gunma University, Japan**


# Outline

- Research Objective
- Introduction to DAC
- Glitches
- Introduction to Gray Code
- Gray Code Input DAC
  - Switch Matrix Design
  - Voltage Mode Gray Code Input DAC
  - Current Steering Mode Gray Code Input DAC
- Conclusion

# Research Objective

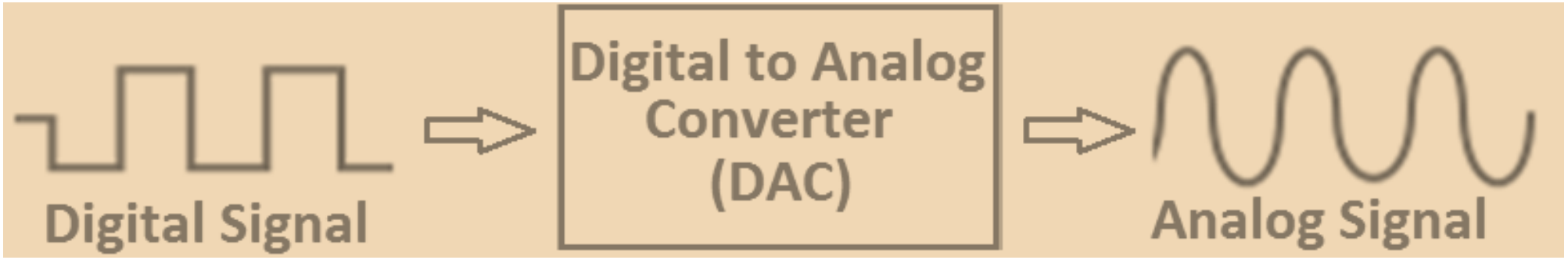
- ✓ Research and implement DAC for glitch reduction using Gray code input
  - *(difficult to design)

Approach

- ✓ Use MOSFETs for DAC design
- ✓ Utilization of Gray code input for glitch reduction

# Introduction to DAC

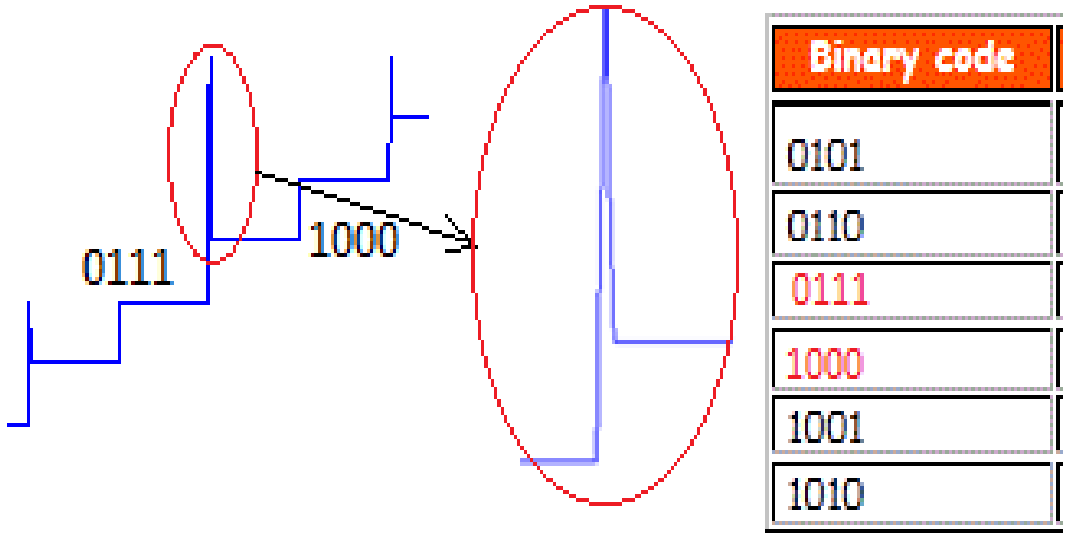
- Convert digital signal to analog signal




- Signal to be recognized by human senses
- Widely used in signal processing

# What are Glitches ?

- ✓ Voltage spikes
- ✓ Reasons for glitches
  - Capacitive coupling
  - Differences in switching




- ➢ Glitch behaviour → Dominated by difference in switching
- ➢ Switching of MSB → Most significant glitches

(Multiple switches changing states at once)

# Glitch Problem and Remedy

## Effects of Glitch

- Serious deterioration of images, videos and sounds

## Remedy

- Using high-order reconstruction filter
- Using track/hold circuitry at the DAC output

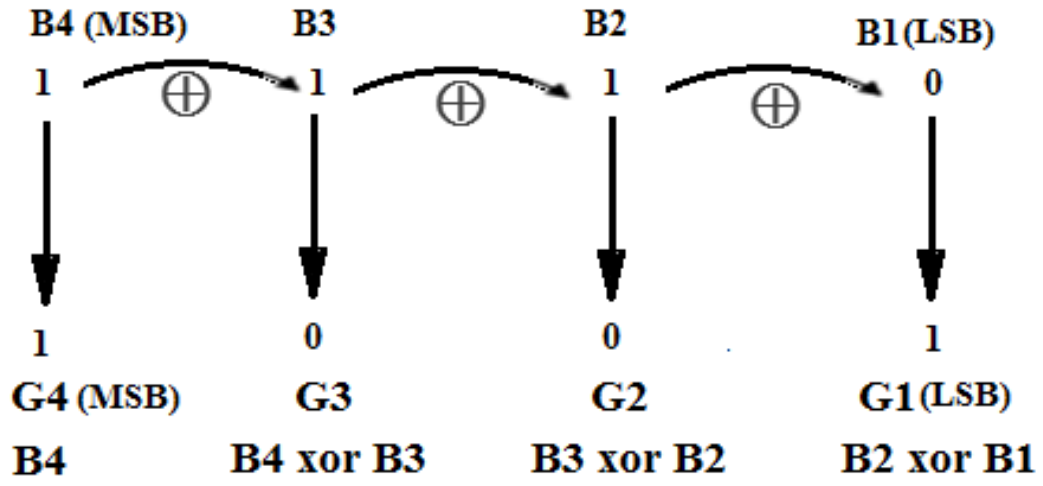
} Extra Space in IC, Expensive

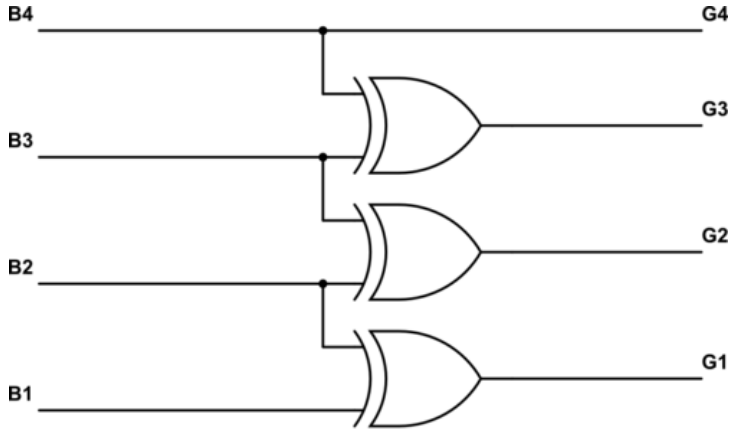
- **Using Gray code input DAC topologies** ✓

# What is Gray code ?

- Gray code → Alternative representation of binary code
- Two adjacent numbers→ Only one bit change
- Reflected binary code

## Binary to Gray code conversion




Binary to Gray code converter

# Gray Code versus Binary Code

**Binary code** Multiple bits change at a time

Trigger more switches

Example. 1 → 2 --- 0001 → 0010 2 bits change

7 → 8--- 0111 →1000 all 4 bits change

**Gray code** Only one bit changes at a time

Triggers one switch

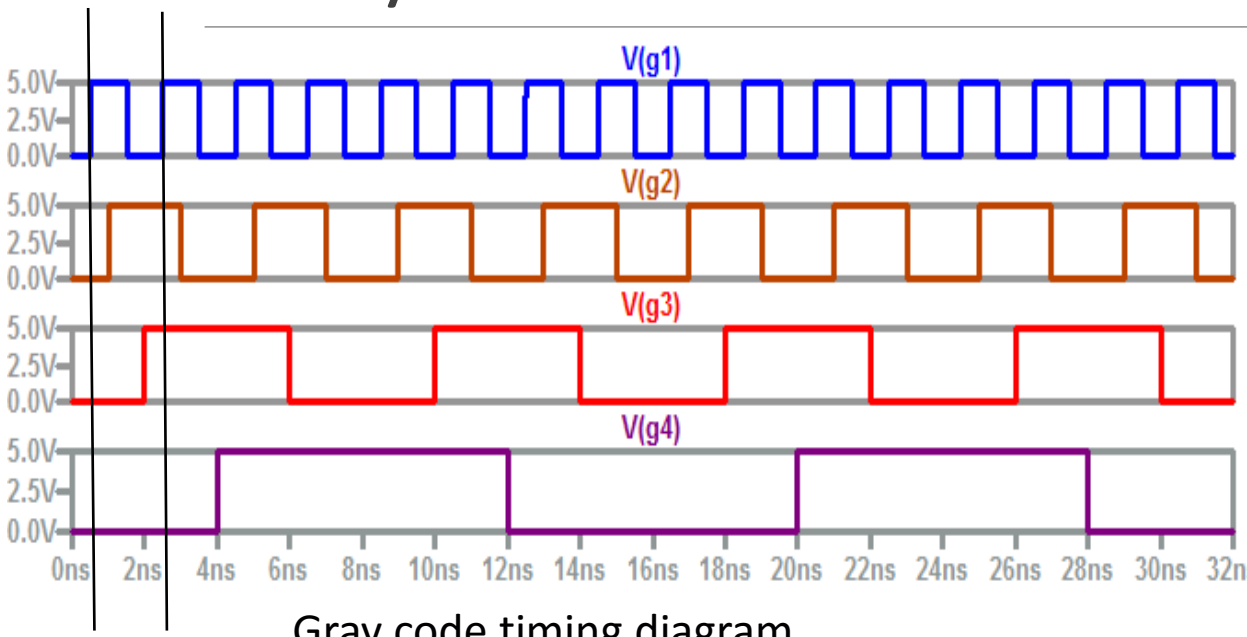
**Example. 1 → 2 --- 0001 → 0011 one bit change (B2)**

**7 → 8 --- 0100→1100 one bit change (B4)**

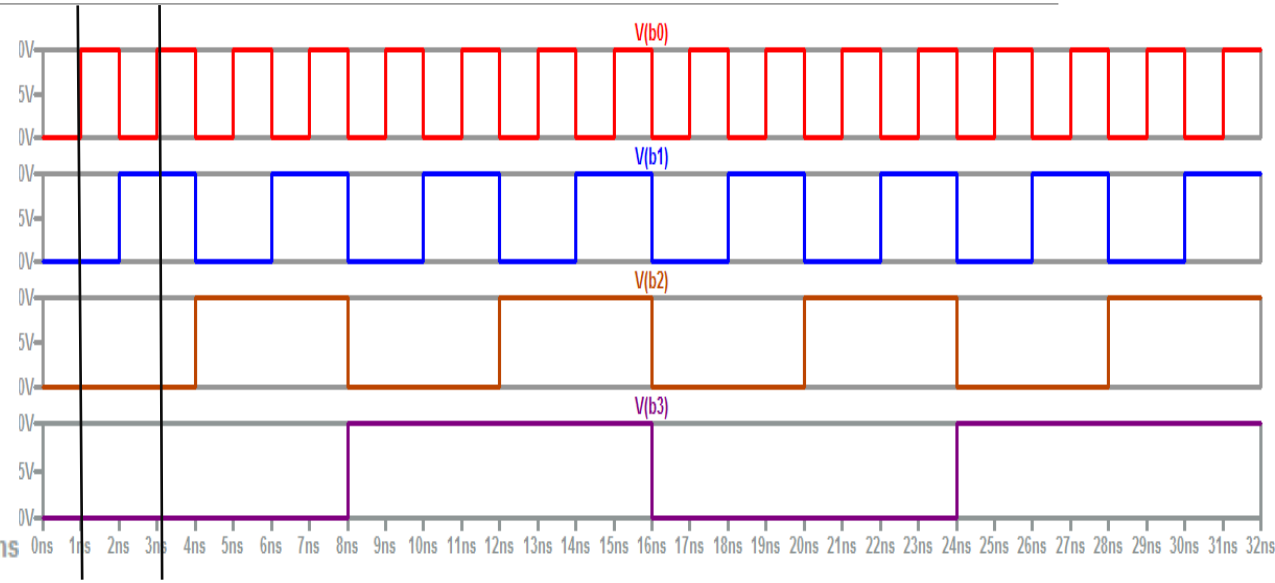
Less glitches

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

# Gray code versus Binary code Timing Diagram




Gray code timing diagram



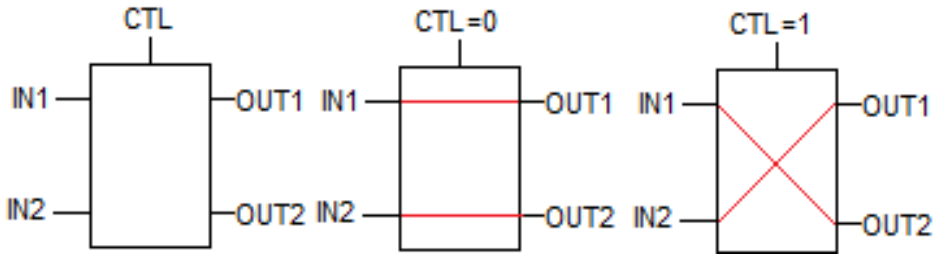

Binary code timing diagram

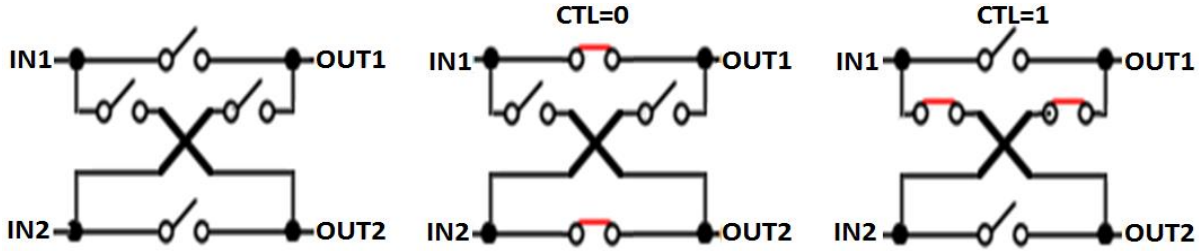
Gray code → Only one bit changes at a time 0001→0011
Binary code → Multiple bits change at a time 0001→0010

Using Gray code → Less glitches expected to appear

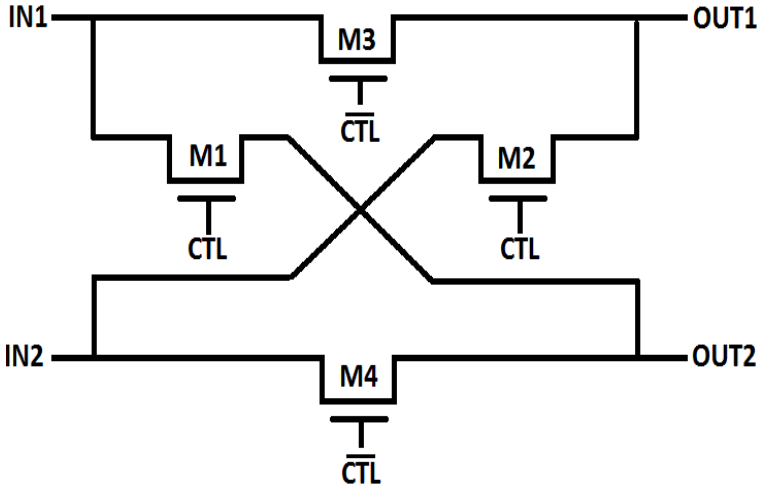
# Gray Code Input DAC

# Switch Matrix Design

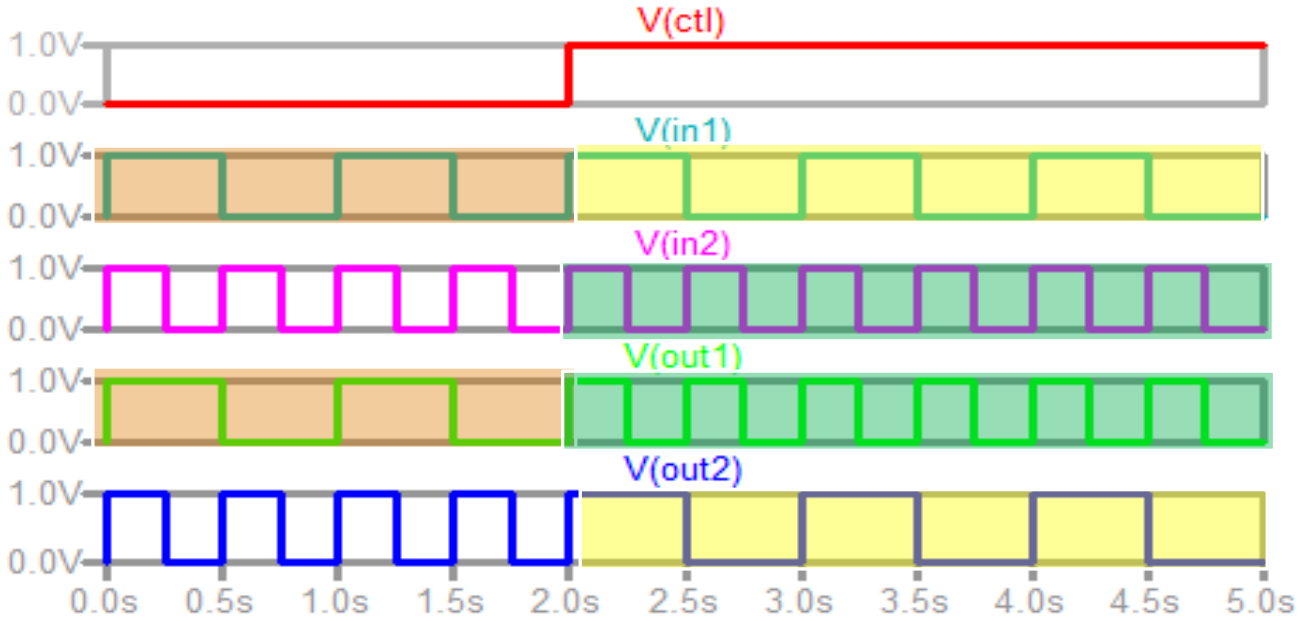


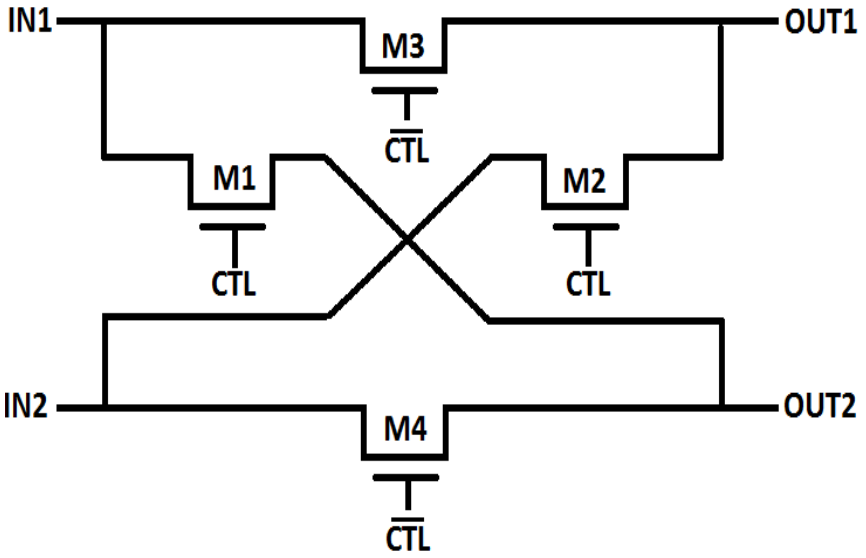




Switch is DPDT (Double-Pole Double-Throw)



# Switch Matrix Operation




CTL → LOW:

M3 , M4 → ON, M1, M2 → OFF

IN1 = OUT1, IN2 = OUT2

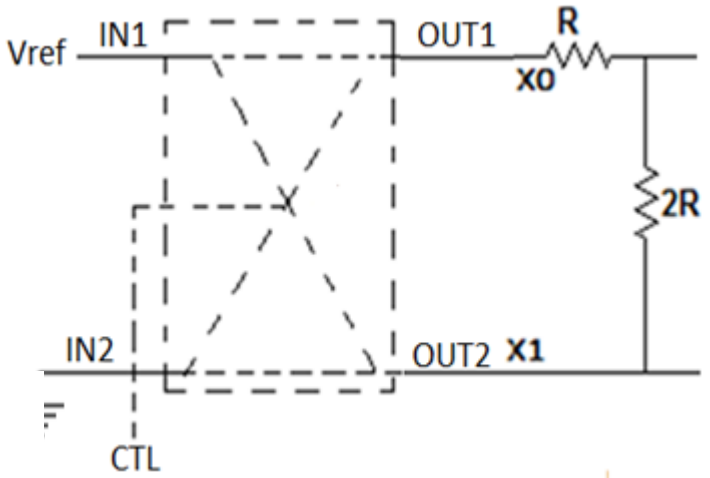
CTL → HIGH:

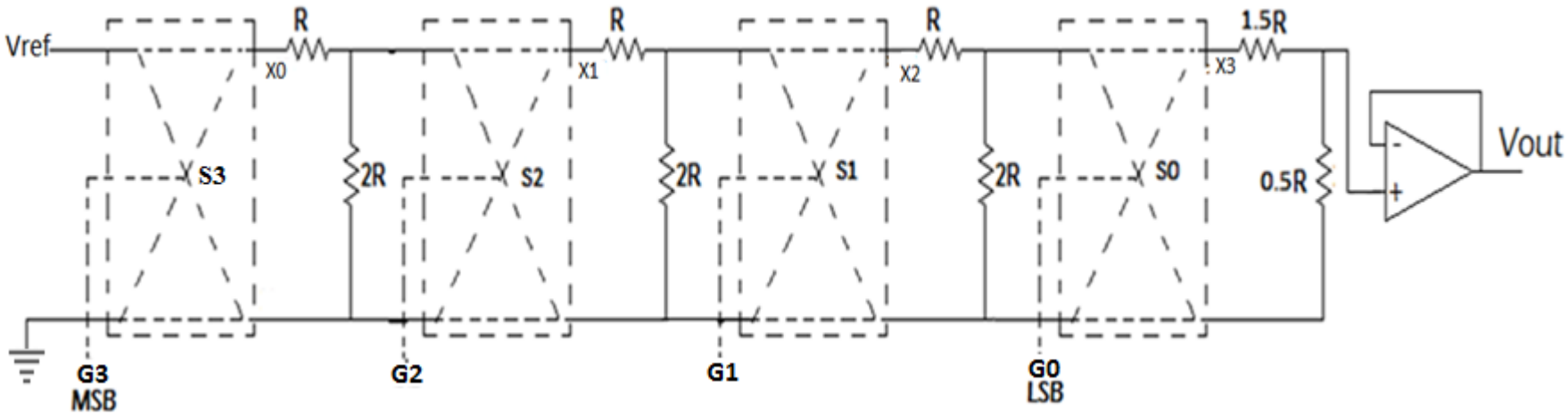
M1, M2 → ON, M3, M4 → OFF

IN1 = Out2 , IN2 =OUT1

# Voltage Mode Gray Code Input DAC

- IN1 = Vref
- IN2 = 0
- CTL ← Gray code input
- OUT1, OUT2 → Connected with R-2R Ladder



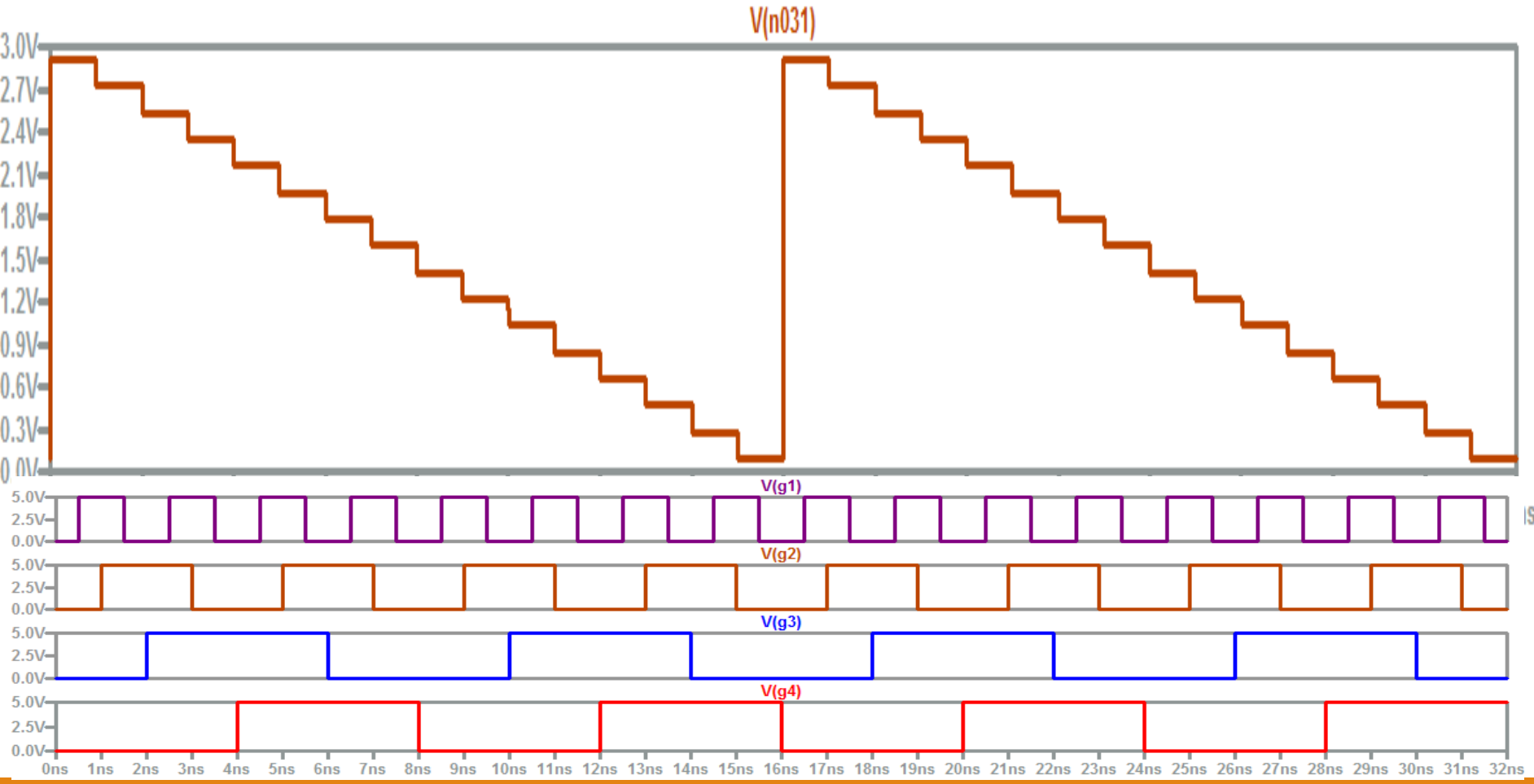

- Final stage → terminated with 1.5R, 0.5R resistors.

$$Vout(D) = \frac{Vref}{2^{n+1}} |(2D - 1)|$$

n : number of bits
D =1, 2, 3...n+1

# Voltage Mode Gray Code Input DAC SPICE Simulation Results

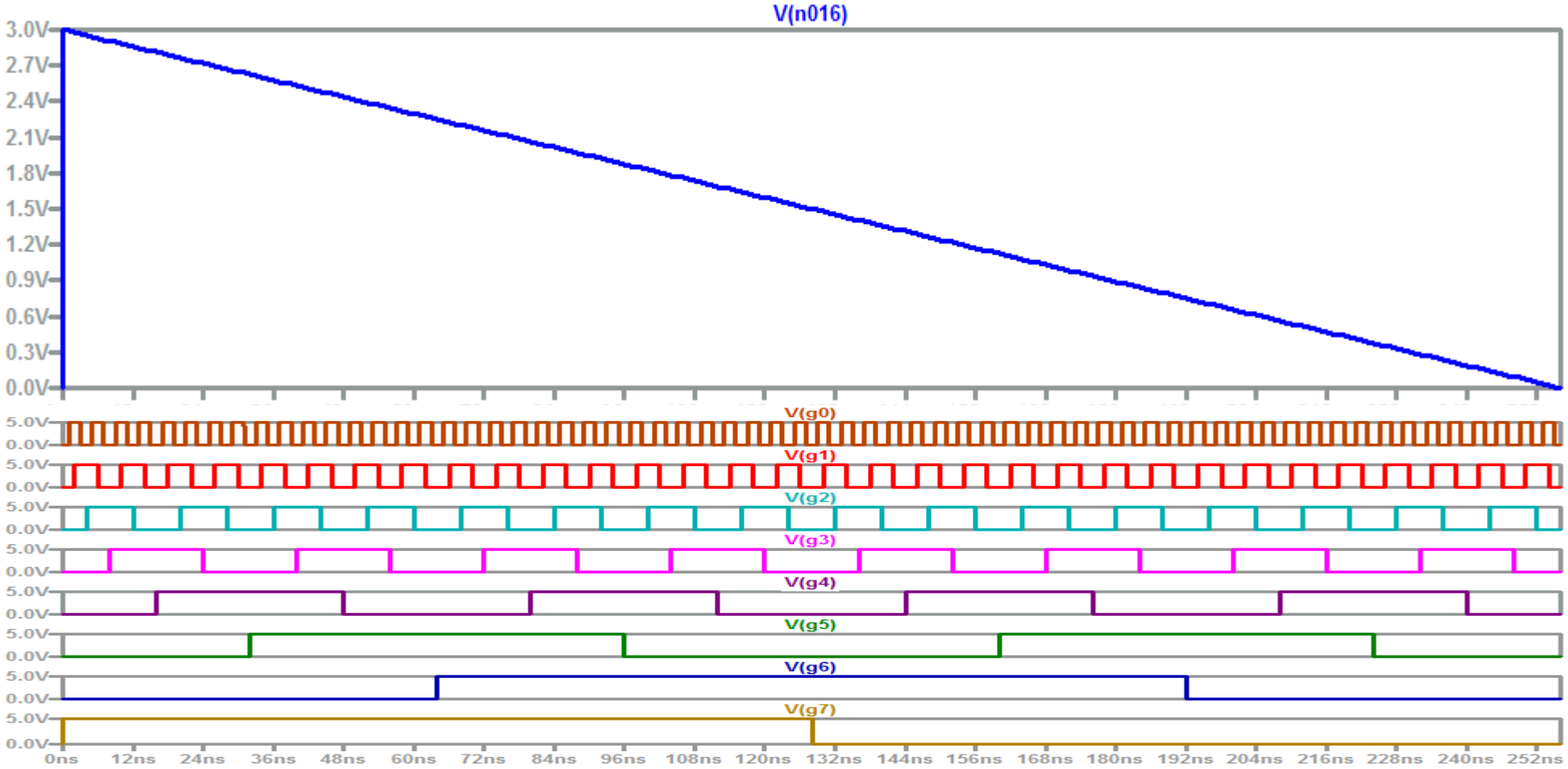
## 4-bit case




| D | Bits | Vout | |
|---|---|---|---|
| 1 | 0000 | 3/32 | 0.09375 |
| 2 | 0001 | 9/32 | 0.28125 |
| 3 | 0011 | 15/32 | 0.46875 |
| 4 | 0010 | 21/32 | 0.65625 |
| 5 | 0110 | 27/32 | 0.84375 |
| 6 | 0111 | 33/32 | 1.03125 |
| 7 | 0101 | 39/52 | 1.21875 |
| 8 | 0100 | 45/32 | 1.40625 |
| 9 | 1100 | 51/32 | 1.59375 |
| 10 | 1101 | 57/32 | 1.78125 |
| 11 | 1111 | 63/32 | 1.96875 |
| 12 | 1110 | 69/32 | 2.15625 |
| 13 | 1010 | 75/32 | 2.34375 |
| 14 | 1011 | 81/32 | 2.53125 |
| 15 | 1001 | 87/32 | 2.71875 |
| 16 | 1000 | 93/32 | 2.90625 |

# Voltage Mode Gray Code Input DAC SPICE Simulation Results

8-bit case

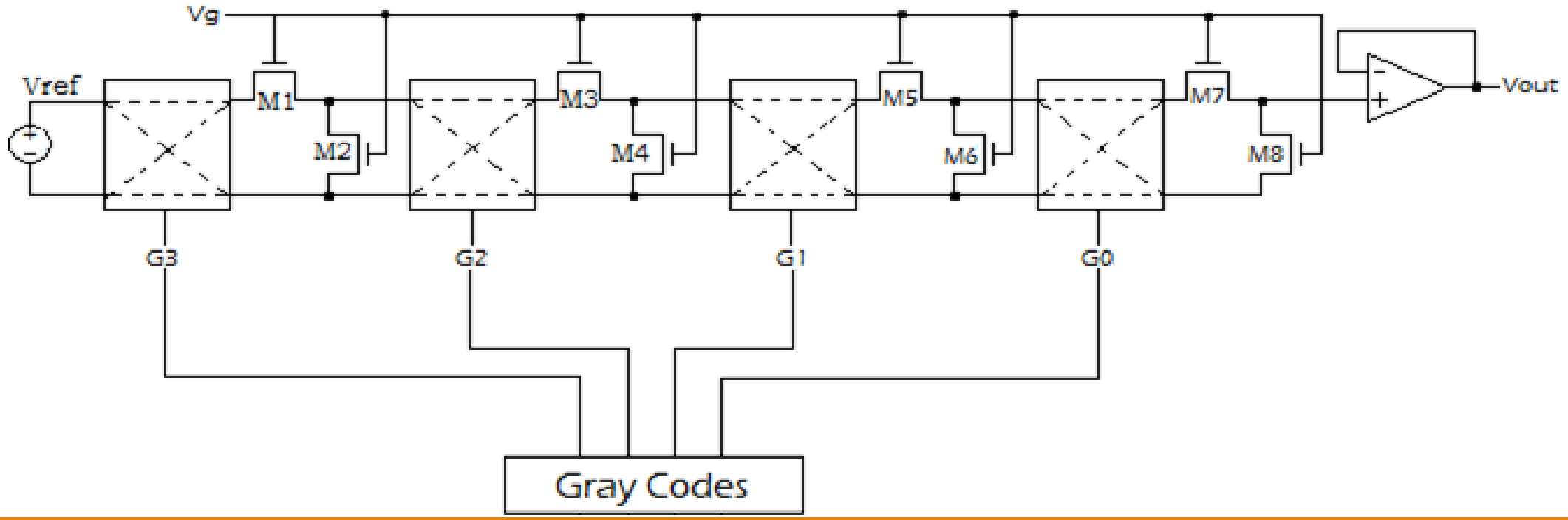



NO GLITCHES

# Voltage Mode Gray Code Input DAC MOSFET Implementation

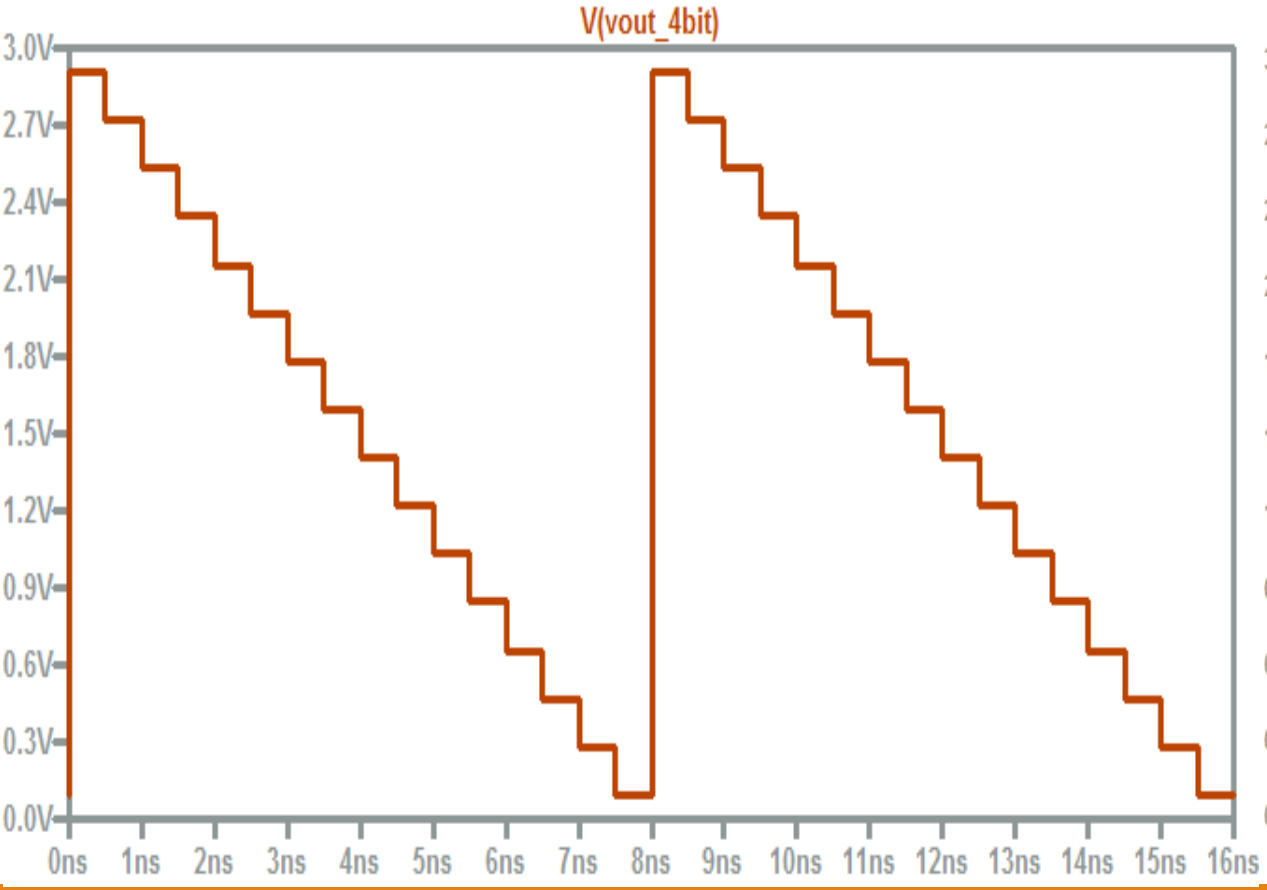
Aspect ratios W/L for R, 2R, 1.5R, 0.5R

$$R = \frac{V_{DS}}{Id_{sat}} = \frac{V_{DS}}{\frac{u_n C_{ox}}{2} \times \left(\frac{W}{L}\right) \times (V_{GS} - V_{TH})^2}$$

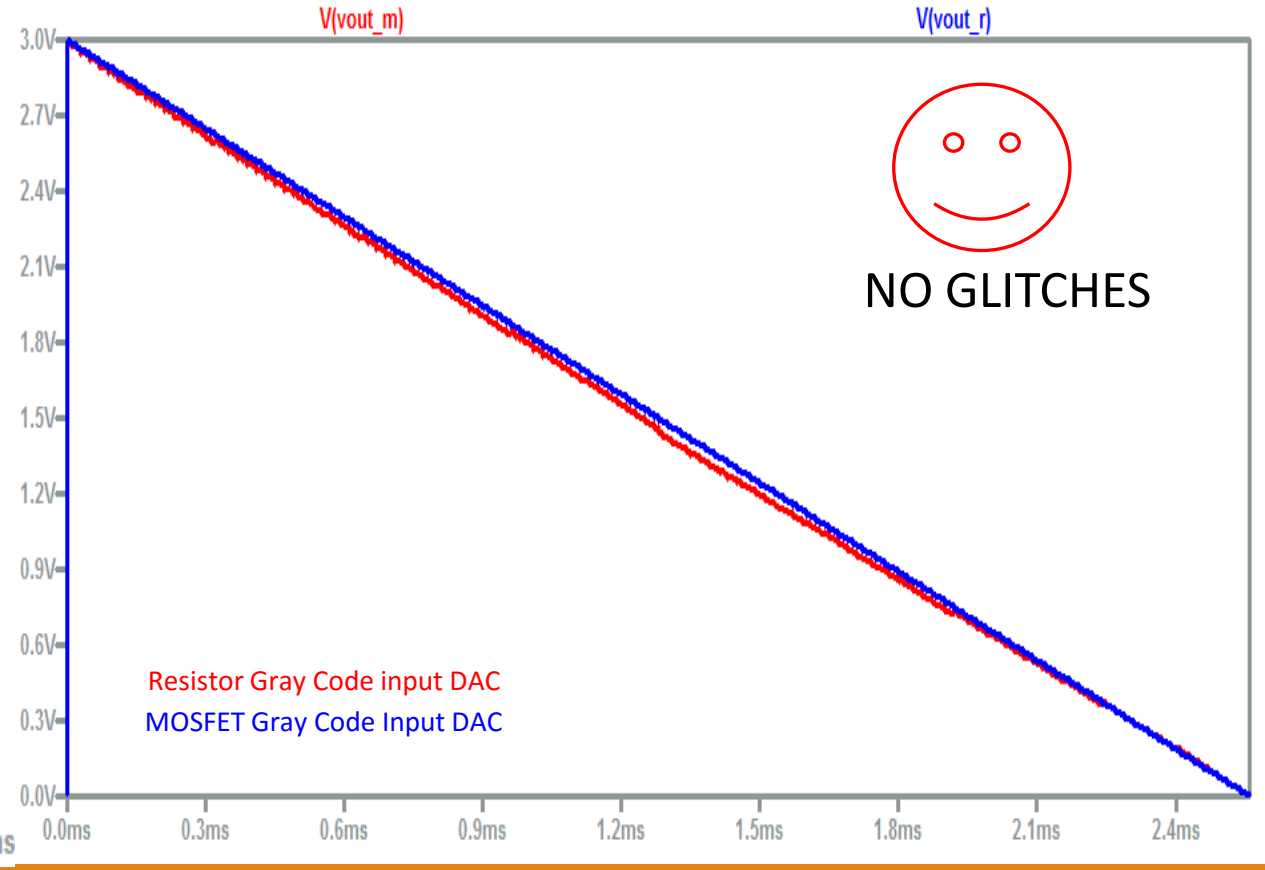


# Voltage Mode Gray Code Input DAC MOSFET Implementation Simulation Results

4-bit case

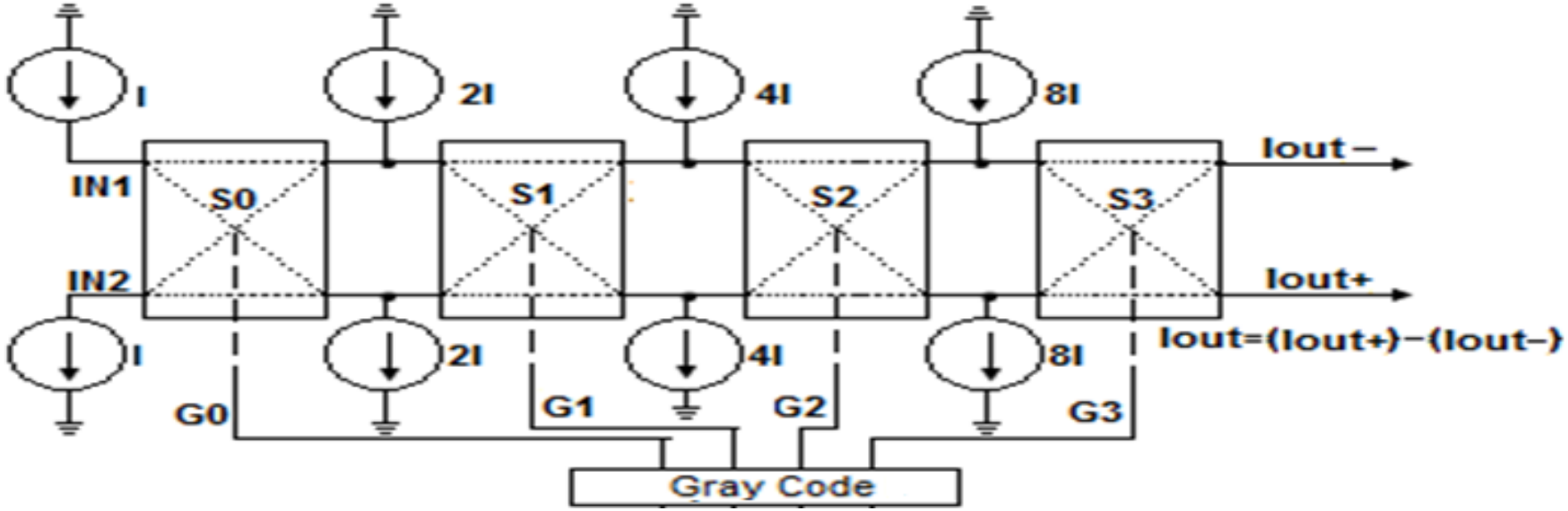



8-bit case



# Current Steering Mode Gray Code Input DAC
# Circuit Configuration

- IN1, IN2, intermediate stages →binary weighted current sources.
- Gray code alters the way the switches are triggered
- Iout=Iout+ – Iout –



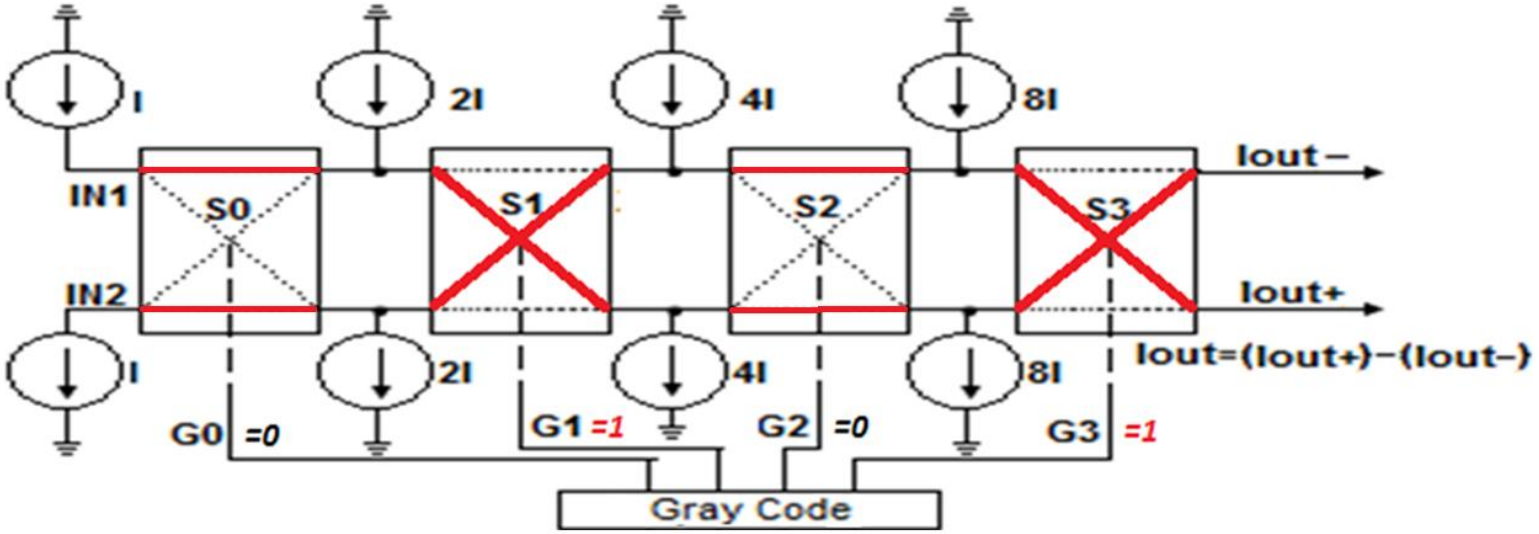
# Current Steering Mode Gray Code Input DAC
## Circuit Operation

For 1010 Gray code,

S3, S1 → ON, the other switches → OFF

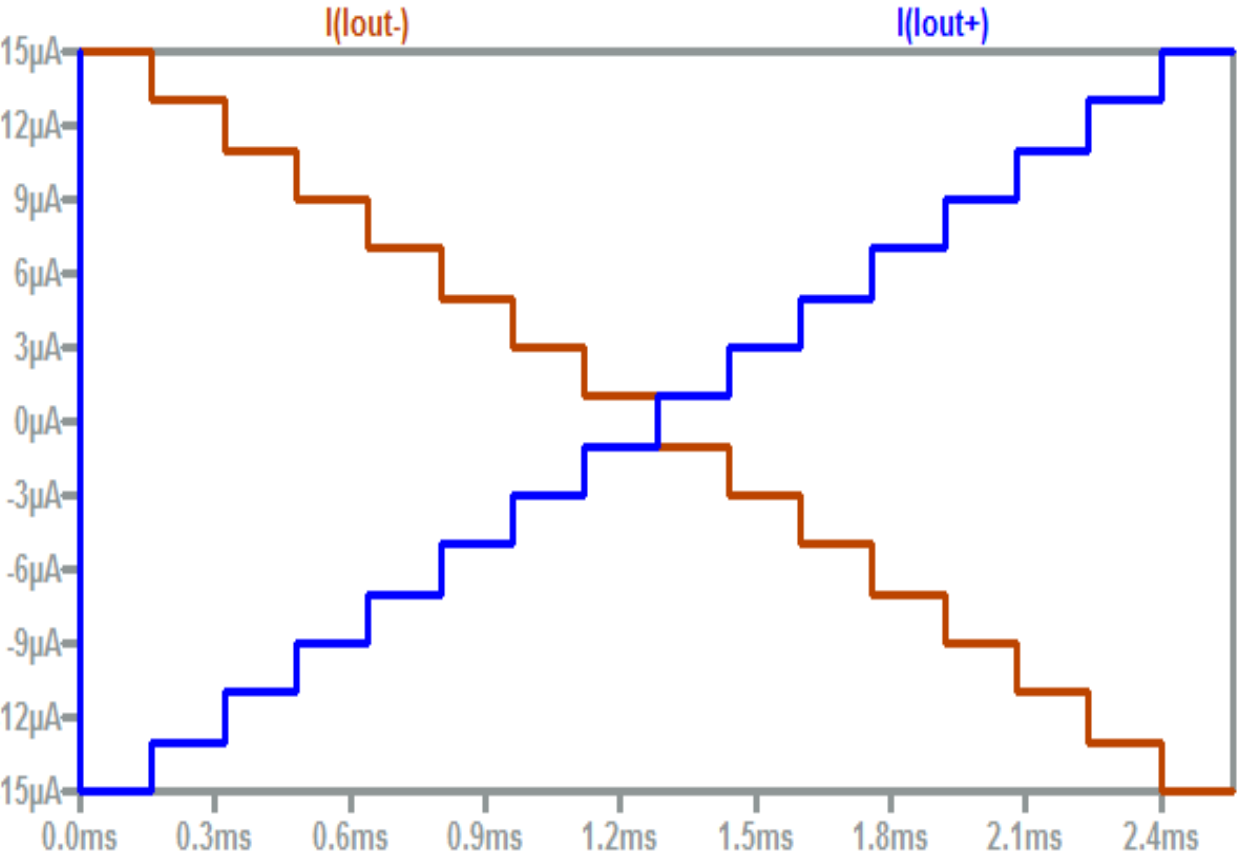
$$Iout -= I + 2I - 4I - 8I = -9I$$

$$Iout += -I - 2I + 4I + 8I = 9I$$

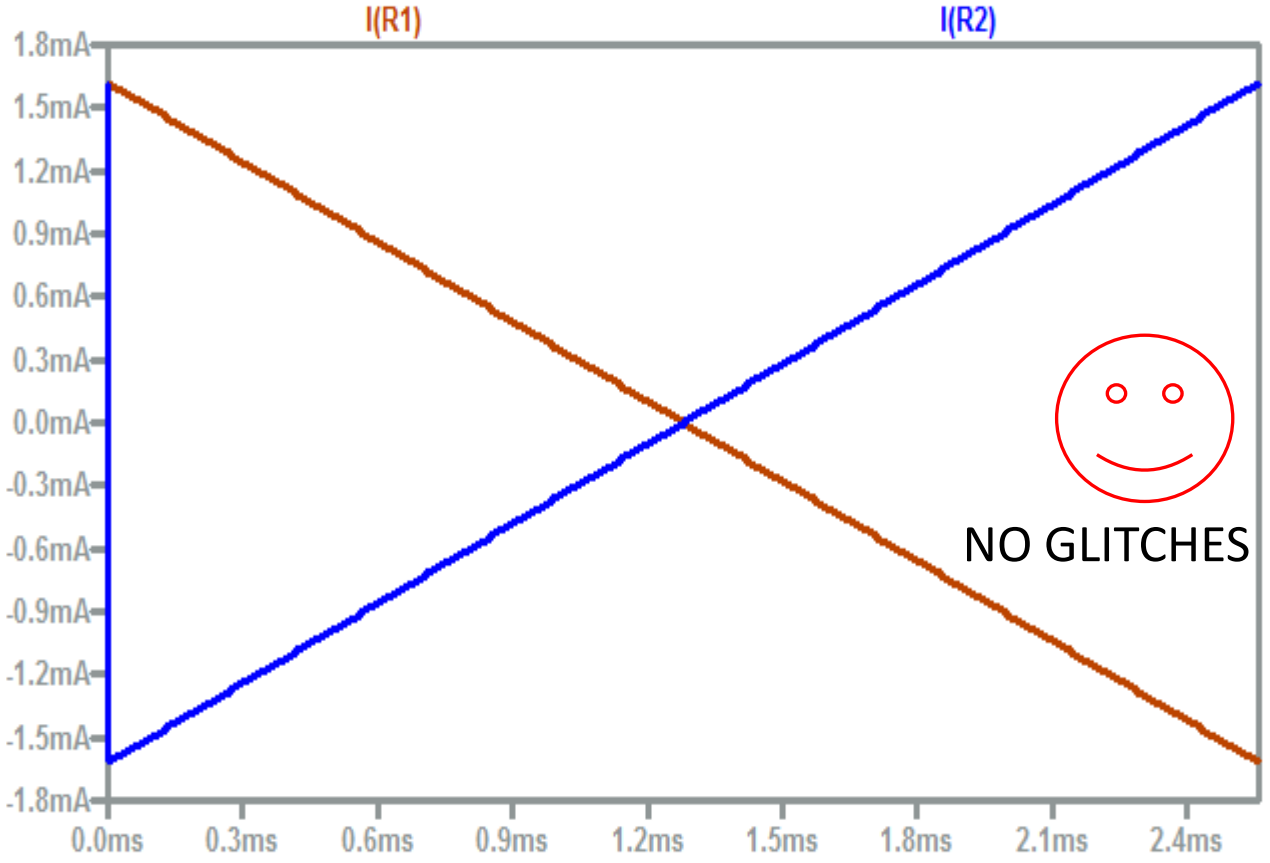


# Current Steering Mode Gray Code Input DAC SPICE Simulation Results

## 4-bit case



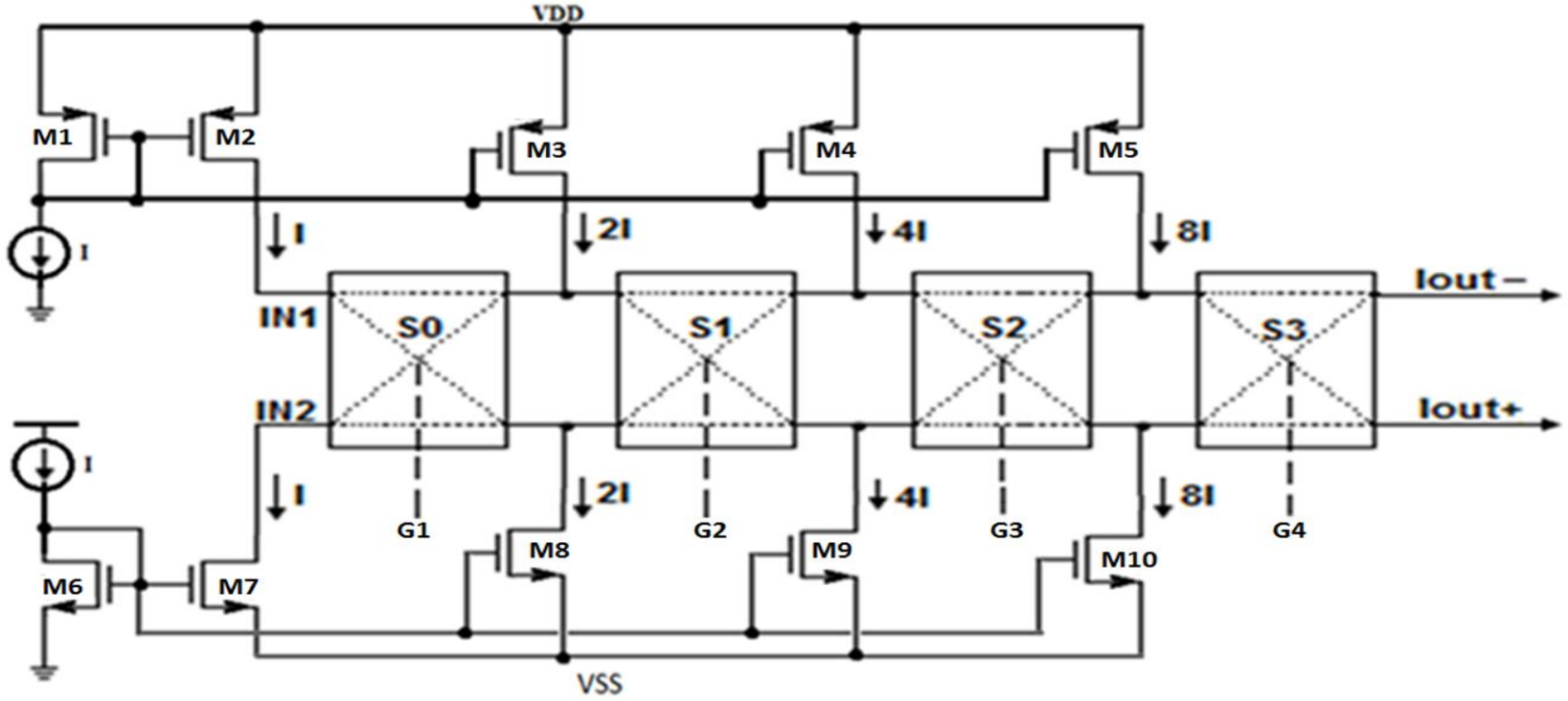

## 8-bit case



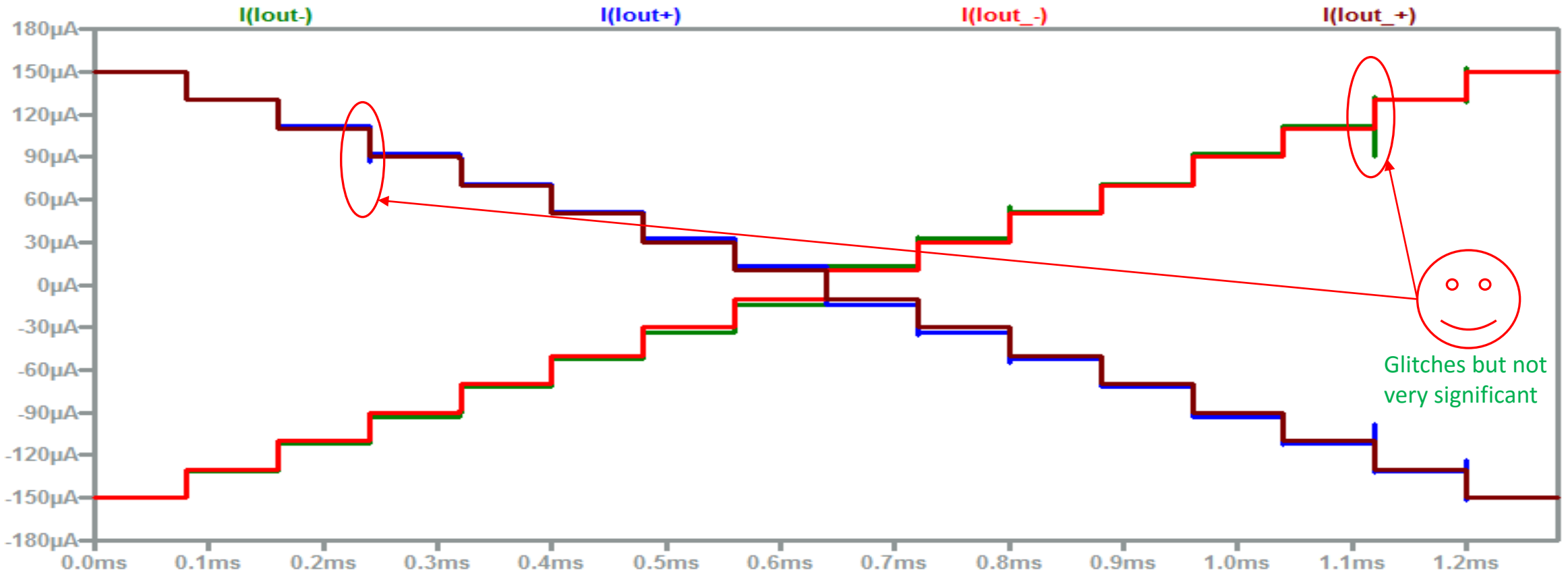
# Current Steering Mode Gray Code Input DAC MOSFET Implementation

$M2,\ M3,\ M4,\ M5$ generate $I,\ 2I,\ 4I, 8I$ (current source)

$M7,\ M8,\ M9, M10$ generate $I,\ 2I,\ 4I, 8I$ (current sink)

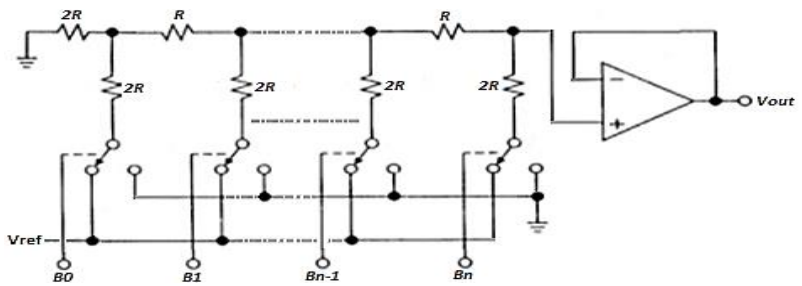


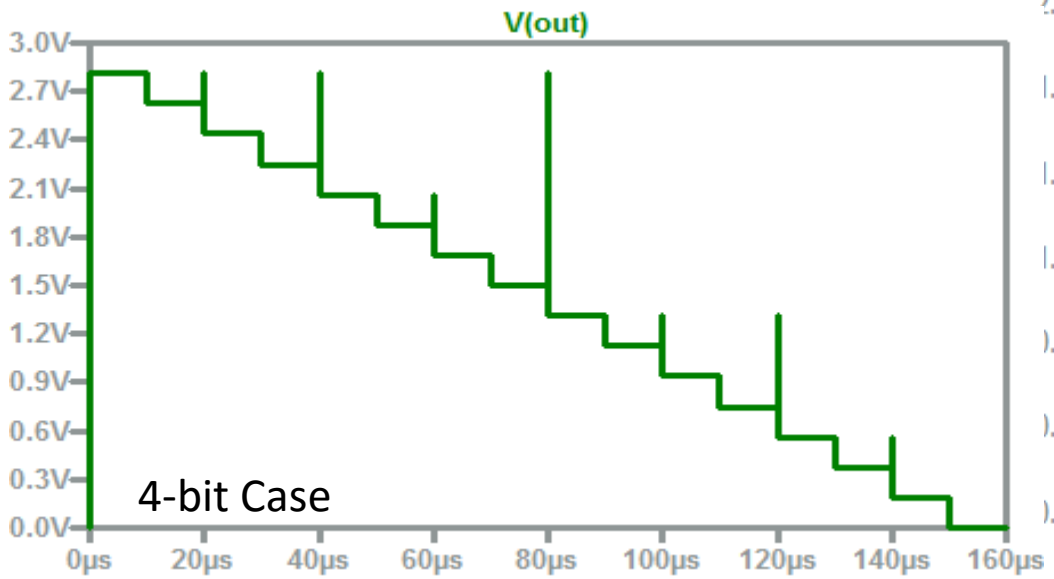
# Current Steering Mode Gray Code Input DAC MOSFET Implementation Simulation Results




These glitches are due to mismatch of PMOS and NMOS

# R-2R Binary Voltage Mode DAC

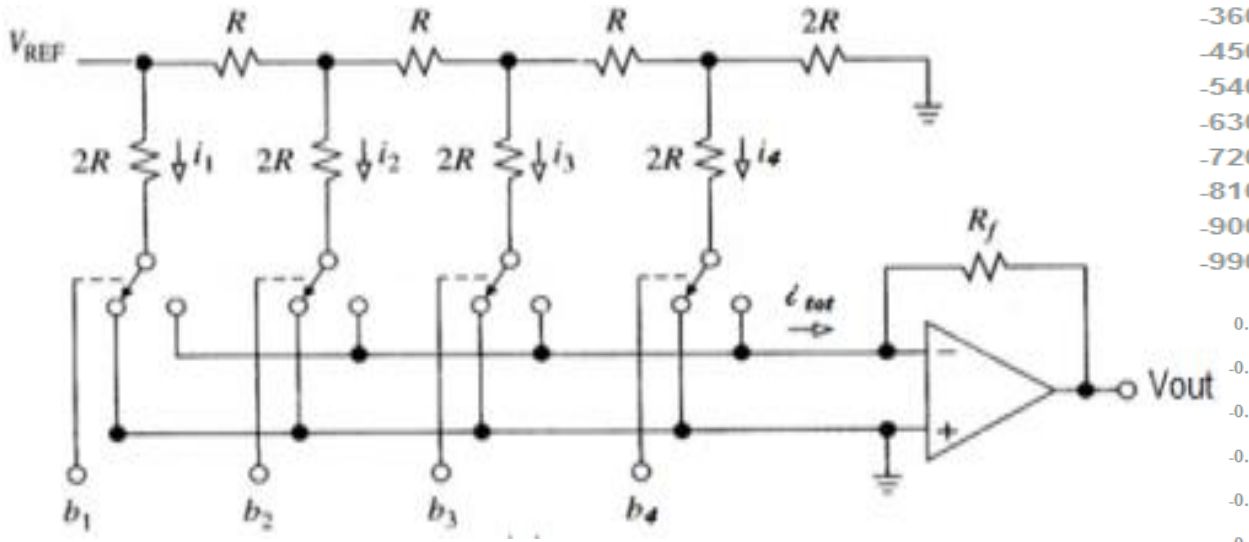


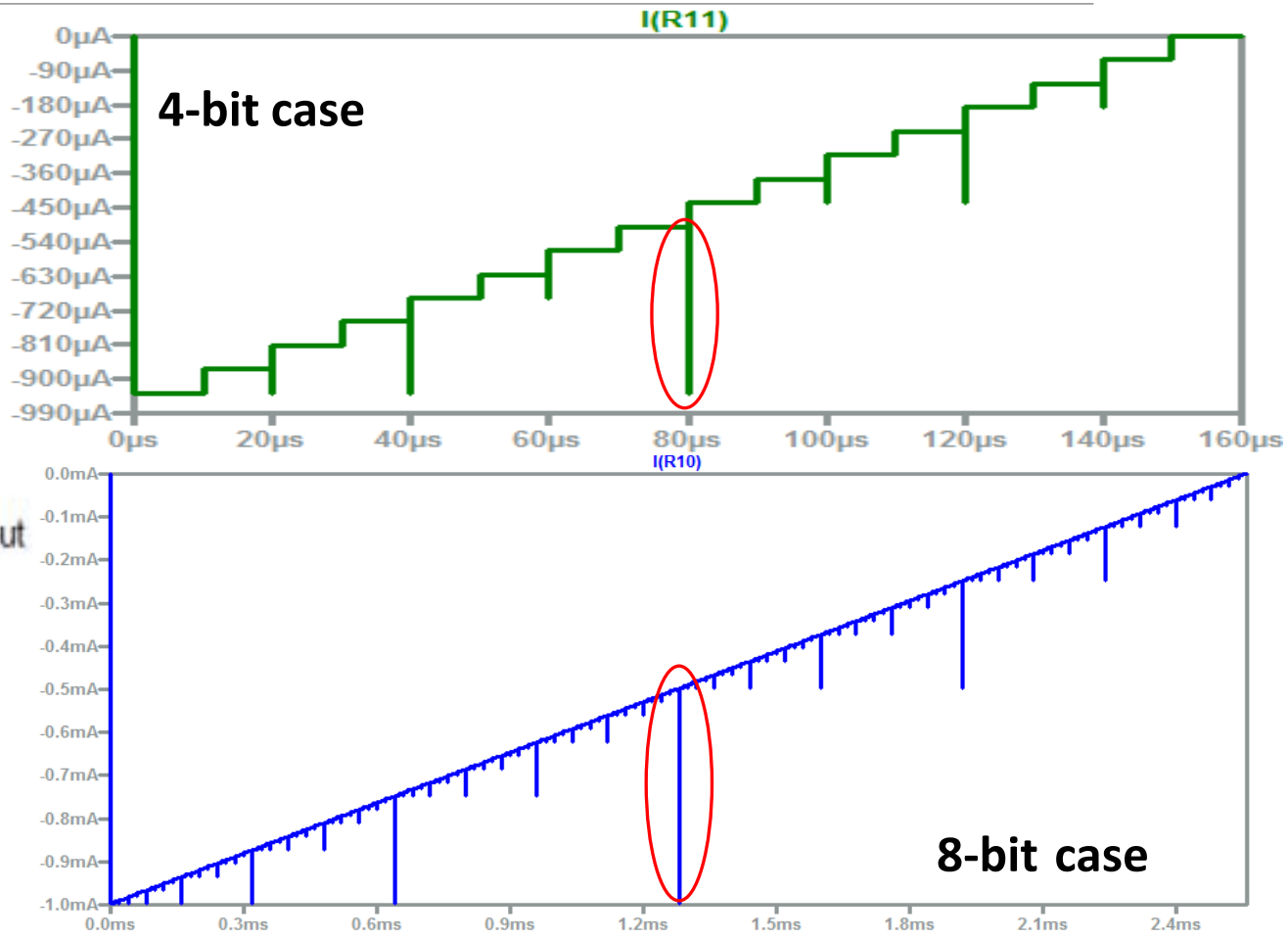





Glitches get introduced in binary code R-2R DAC

# R-2R Binary Current Mode DAC



# Conclusion

- ✓ R-2R DACs prone to glitches → Multiple bits switching at a time.
- ✓ Claims of Gray code DACs being difficult to design

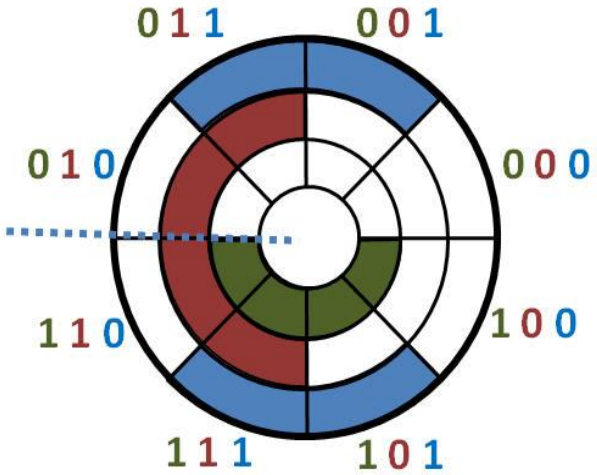
  but successfully designed and simulated
  - ✓ Gray code Input DACs reduce glitches considerably
  - ✓ No extra space needed for IC
  - ✓ No extra circuit needed to remove glitches

# Final Statement

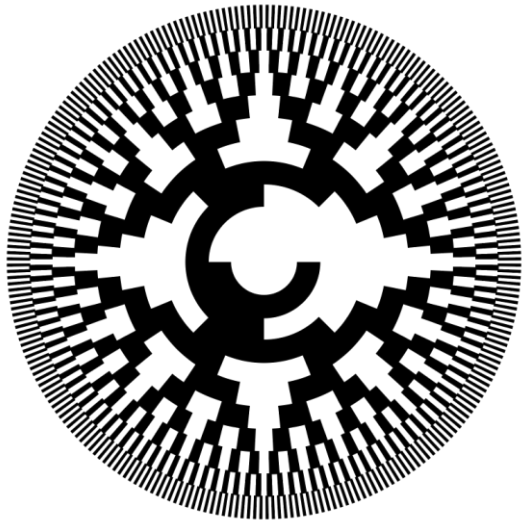
Coding method can lead to robust mixed-signal circuit design.

Gray code was invented by Frank Gray at Bell Lab in 1947.


FRANK GRAY and A. L. Johnsrud in television booth. Behind the glass panels at sides and top are the photo-electric cells.




| $Q_2Q_1Q_0$ |
|---|
| 000 |
| 001 |
| 011 |
| 010 |
| 110 |
| 111 |
| 101 |
| 100 |



# Thank you