Immink Codes for Phase Change Memory

Hiroshi Kamabe

Gifu University

June 15, 2017

Figure 1 by Shannon

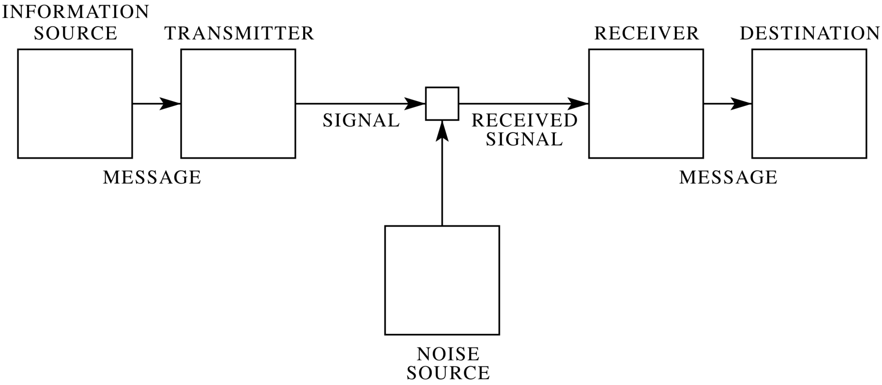


Fig. 1—Schematic diagram of a general communication system.

C. Shannon, "A mathematical theory of communication", The Bell system Technical Journal, vol. 27, pp.379–423, 623–656, July, October, 1948.

K. Immink's talk at Almaden research center, CA, USA

Figure 1 by Shannon

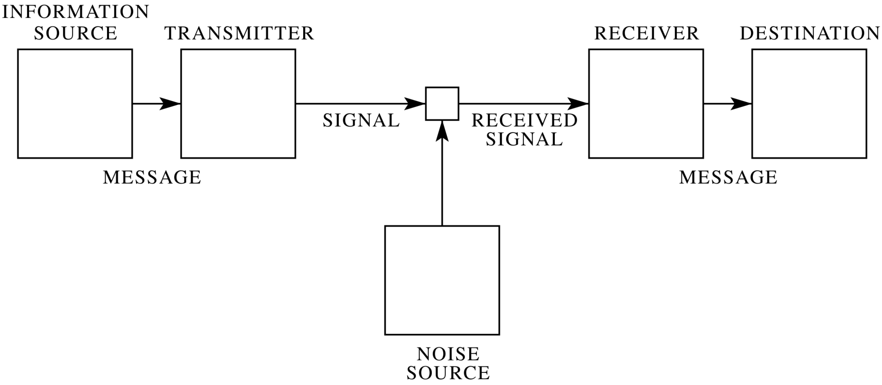


Fig. 1—Schematic diagram of a general communication system.

C. Shannon, "A mathematical theory of communication", The Bell system Technical Journal, vol. 27, pp.379–423, 623–656, July, October, 1948.

K. Immink's talk at Almaden research center, CA, USA.

Figure 1 by Shannon(in Japanese)

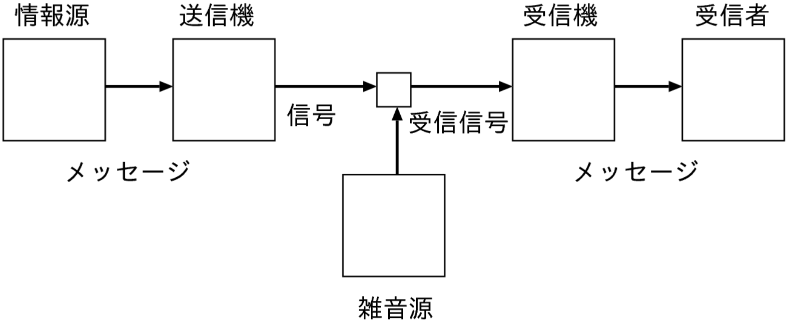


Figure: 一般的な通信路の概念図

C. Shannon, "A mathematical theory of communication", The Bell system Technical Journal, vol. 27, pp.379–423, 623–656, July, October, 1948.

Digital Communication System

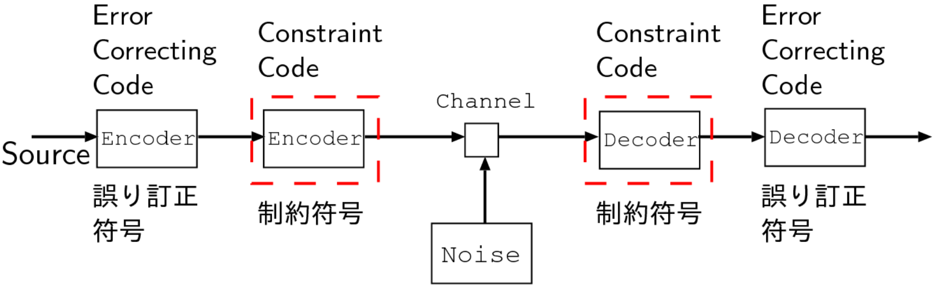
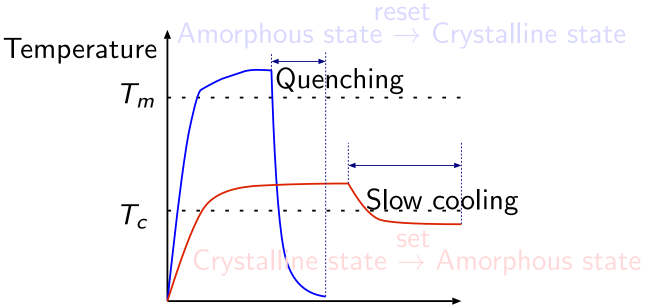


Figure: Schematic Diagram of Digital Communication System

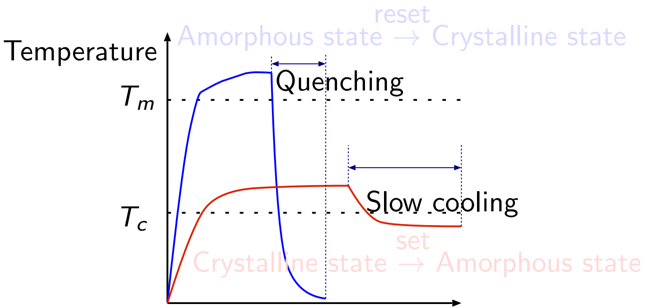
Introduction

- Phase change memory: a non volatile semiconductor memory to which we can store digital data almost freely(no restriction, i.e., the number of rewrites of flash memory cells is almost ∞, etc).
- The phase (or state) of a cell of PCM is changed by heating; quick heating and cooling (amorphous to crystalline state, reset operation), slow heating and cooling (crystalline to amorphous state, set operation)
- Amorphous state: High resistance,1 , Crystalline state:Low resistance, 0



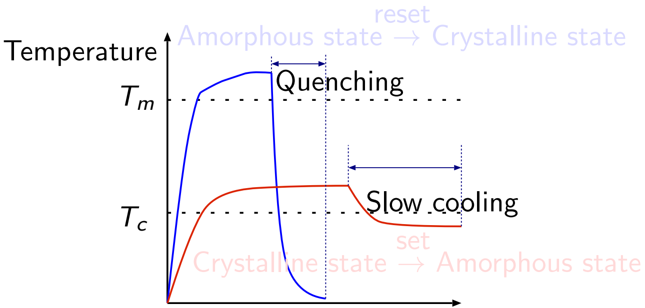
Introduction

- Phase change memory: a non volatile semiconductor memory to which we can store digital data almost freely(no restriction, i.e., the number of rewrites of flash memory cells is almost ∞, etc).
- The phase (or state) of a cell of PCM is changed by heating; quick heating and cooling (amorphous to crystalline state, reset operation), slow heating and cooling (crystalline to amorphous state, set operation)
- Amorphous state: High resistance,1 , Crystalline state:Low resistance, 0



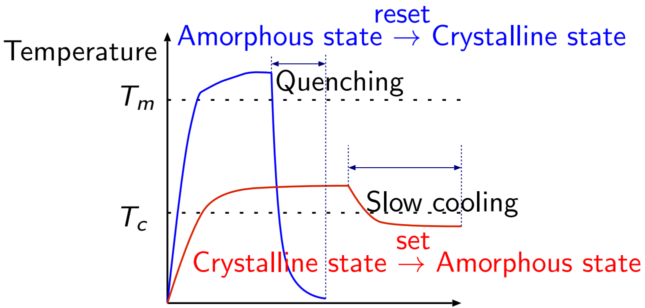
Introduction

- Phase change memory: a non volatile semiconductor memory to which we can store digital data almost freely(no restriction, i.e., the number of rewrites of flash memory cells is almost ∞, etc).
- The phase (or state) of a cell of PCM is changed by heating; quick heating and cooling (amorphous to crystalline state, reset operation), slow heating and cooling (crystalline to amorphous state, set operation)
- Amorphous state: High resistance,1 , Crystalline state:Low resistance, 0

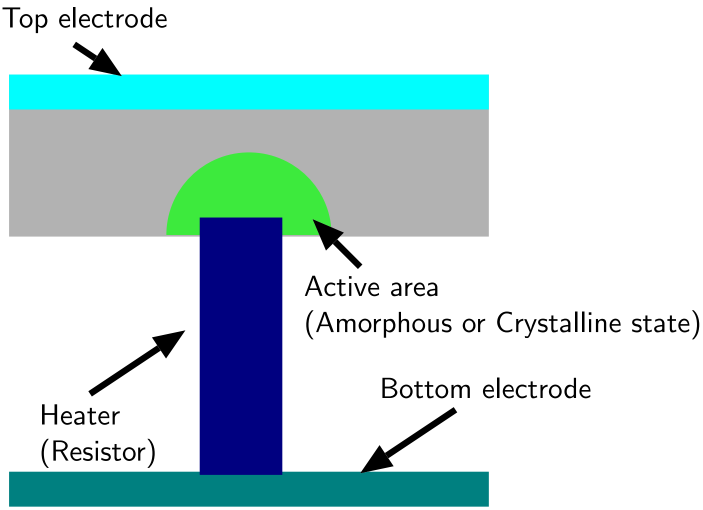


Introduction

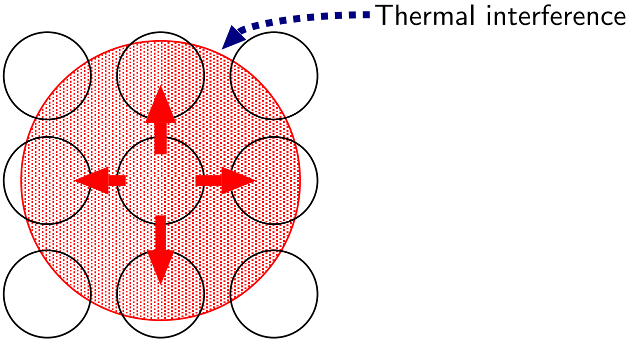
- Phase change memory: a non volatile semiconductor memory to which we can store digital data almost freely(no restriction, i.e., the number of rewrites of flash memory cells is almost ∞, etc).
- The phase (or state) of a cell of PCM is changed by heating; quick heating and cooling (amorphous to crystalline state, reset operation), slow heating and cooling (crystalline to amorphous state, set operation)
- Amorphous state: High resistance,1 , Crystalline state:Low resistance, 0



Cell Structure of PCM

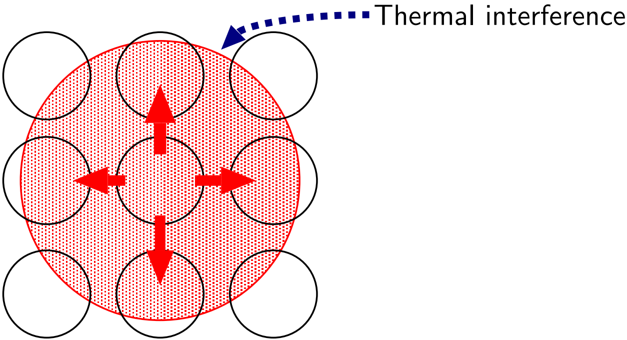


Cross talk problem, set and reset operation



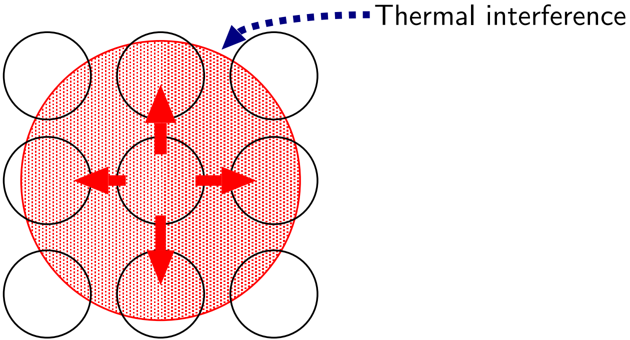
- The reset operation causes a strong thermal interference.
- A 2-dimensional constraint is needed but the capacity of the 2-dimensional constraint is small in many cases.
- We construct a 1-dimensional constraint code and evaluate it by computer simulation.

Cross talk problem, set and reset operation



- The reset operation causes a strong thermal interference.
- A 2-dimensional constraint is needed but the capacity of the 2-dimensional constraint is small in many cases.
- We construct a 1-dimensional constraint code and evaluate it by computer simulation.

Cross talk problem, set and reset operation



- The reset operation causes a strong thermal interference.
- A 2-dimensional constraint is needed but the capacity of the 2-dimensional constraint is small in many cases.
- We construct a 1-dimensional constraint code and evaluate it by computer simulation.

Our Code

1. K. Cai constructed an efficient code for k-constraint where $k = 3$ by using Immink coding [K.Cai, 2014].
2. We also construct a code for $k = 3$ to avoid long runs of zeros.
3. When $k = 3$, left ends of k constrained sequences are

$$1, \quad ,01, \quad 001, \quad 0001.$$

4. Let s_1, s_2, s_3 and s_4 be numbers of states corresponding to the above sequences, respectively.
5. Then apply the Immink coding.
6. Tune the resulting code carefully.

Our Code

1. K. Cai constructed an efficient code for k-constraint where $k = 3$ by using Immink coding [K.Cai, 2014].
2. We also construct a code for $k = 3$ to avoid long runs of zeros.
3. When $k = 3$, left ends of k constrained sequences are

$$1, \quad ,01, \quad 001, \quad 0001.$$

4. Let s_1, s_2, s_3 and s_4 be numbers of states corresponding to the above sequences, respectively.
5. Then apply the Immink coding.
6. Tune the resulting code carefully.

Our Code

1. K. Cai constructed an efficient code for k-constraint where $k = 3$ by using Immink coding [K.Cai, 2014].
2. We also construct a code for $k = 3$ to avoid long runs of zeros.
3. When $k = 3$, left ends of k constrained sequences are

$$1, \quad ,01, \quad 001, \quad 0001.$$

4. Let s_1, s_2, s_3 and s_4 be numbers of states corresponding to the above sequences, respectively.
5. Then apply the Immink coding.
6. Tune the resulting code carefully.

Our Code

1. K. Cai constructed an efficient code for k-constraint where $k = 3$ by using Immink coding [K.Cai, 2014].
2. We also construct a code for $k = 3$ to avoid long runs of zeros.
3. When $k = 3$, left ends of k constrained sequences are

$$1, \qquad ,01, \qquad 001, \qquad 0001.$$

4. Let s_1, s_2, s_3 and s_4 be numbers of states corresponding to the above sequences, respectively.
5. Then apply the Immink coding.
6. Tune the resulting code carefully.

Our Code

1. K. Cai constructed an efficient code for k-constraint where $k = 3$ by using Immink coding [K.Cai, 2014].
2. We also construct a code for $k = 3$ to avoid long runs of zeros.
3. When $k = 3$, left ends of k constrained sequences are

$$1, \qquad , 01, \qquad 001, \qquad 0001.$$

4. Let s_1, s_2, s_3 and s_4 be numbers of states corresponding to the above sequences, respectively.
5. Then apply the Immink coding.
6. Tune the resulting code carefully.

Our Code

1. K. Cai constructed an efficient code for k-constraint where $k = 3$ by using Immink coding [K.Cai, 2014].
2. We also construct a code for $k = 3$ to avoid long runs of zeros.
3. When $k = 3$, left ends of k constrained sequences are

$$1, \qquad ,01, \qquad 001, \qquad 0001.$$

4. Let s_1, s_2, s_3 and s_4 be numbers of states corresponding to the above sequences, respectively.
5. Then apply the Immink coding.
6. Tune the resulting code carefully.

Immink coding for $d = 1$ constraint

- n: code length
- E_{ab}: set of constrained sequences of length n starting with symbol a and terminating with symbol b.

1. Find positive integers r_1 and r_2 satisfying the following inequalities:

$$(r_1 + r_2)|E_{00}| + r_1|E_{01}| \geq r_1 2^m \quad (1)$$

$$(r_1 + r_2)(|E_{00}| + |E_{10}|) + r_1(|E_{01}| + |E_{11}|) \geq (r_1 + r_2)2^m \quad (2)$$

2. If we can find the r_1 and r_2, then according to the we can construct an encoding rule and a corresponding decoding rule for the $d = 1$ constraint.
3. Note: When we construct these rules, we can almost ignore constrained patterns and can concentrate only on the numbers of patterns, $|E_{00}|, \ldots$

Immink coding for $d = 1$ constraint

- n: code length
- E_{ab}: set of constrained sequences of length n starting with symbol a and terminating with symbol b.

1. Find positive integers r_1 and r_2 satisfying the following inequalities:

$$
\begin{aligned}
(r_1 + r_2)|E_{00}| + r_1|E_{01}| &\geq r_1 2^m & (1)\\
(r_1 + r_2)(|E_{00}| + |E_{10}|) + r_1(|E_{01}| + |E_{11}|) &\geq (r_1 + r_2)2^m & (2)
\end{aligned}
$$

2. If we can find the r_1 and r_2, then according to the we can construct an encoding rule and a corresponding decoding rule for the $d = 1$ constraint.
3. Note: When we construct these rules, we can almost ignore constrained patterns and can concentrate only on the numbers of patterns, $|E_{00}|, \ldots$

Immink coding for $d = 1$ constraint

- n: code length
- E_{ab}: set of constrained sequences of length n starting with symbol a and terminating with symbol b.

1. Find positive integers r_1 and r_2 satisfying the following inequalities:

$$(r_1 + r_2)|E_{00}| + r_1|E_{01}| \geq r_1 2^m \quad (1)$$

$$(r_1 + r_2)(|E_{00}| + |E_{10}|) + r_1(|E_{01}| + |E_{11}|) \geq (r_1 + r_2)2^m \quad (2)$$

2. If we can find the r_1 and r_2, then according to the we can construct an encoding rule and a corresponding decoding rule for the $d = 1$ constraint.

3. Note: When we construct these rules, we can almost ignore constrained patterns and can concentrate only on the numbers of patterns, $|E_{00}|, \ldots$

Immink coding for $d = 1$ constraint

- n: code length
- E_{ab}: set of constrained sequences of length n starting with symbol a and terminating with symbol b.

1. Find positive integers r_1 and r_2 satisfying the following inequalities:

$$(r_1 + r_2)|E_{00}| + r_1|E_{01}| \geq r_1 2^m \quad (1)$$

$$(r_1 + r_2)(|E_{00}| + |E_{10}|) + r_1(|E_{01}| + |E_{11}|) \geq (r_1 + r_2)2^m \quad (2)$$

2. If we can find the r_1 and r_2, then according to the
we can construct an encoding rule and a corresponding decoding rule for the $d = 1$ constraint.
3. Note: When we construct these rules, we can almost ignore constrained patterns and can concentrate only on the numbers of patterns, $|E_{00}|, \ldots$

Our Code

1. K. Cai constructed an efficient code for k-constraint where $k = 3$ by using Immink coding [K.Cai, 2014].
2. We also construct a code for $k = 3$ to avoid long runs of zeros.
3. When $k = 3$, left ends of k constrained sequences are

$$1, \qquad ,01, \qquad 001, \qquad 0001.$$

4. Let s_1, s_2, s_3 and s_4 be numbers of states corresponding to the above sequences, respectively.
5. Then apply the Immink coding.
6. Tune the resulting code carefully.

Step 1

- If we can find positive integers s_1, s_2, s_3 and s_4 and m satisfying the following equations, we can construct a code by Immink coding [K. Cai 2014].

$$
\begin{aligned}
(s_1+s_2+s_3+s_4)|C_{1\cdots1}| &\geq s_1 2^m \\
(s_1+s_2+s_3+s_4)(|C_{1\cdots1}|+|C_{01\cdots1}|) &\geq (s_1+s_2)2^m \\
(s_1+s_2+s_3+s_4)(|C_{1\cdots1}|+|C_{01\cdots1}|+|C_{001\cdots1}|) & \\
+(s_1+s_2+s_3)(|C_{1\cdots10}|+|C_{01\cdots10}|) &\geq (s_1+s_2+s_3)2^m \\
(s_1+s_2+s_3+s_4)(|C_{1\cdots1}|+|C_{01\cdots1}|+|C_{001\cdots1}|+|C_{0001\cdots1}|) & \\
+(s_1+s_2+s_3)(|C_{1\cdots10}|+|C_{01\cdots10}|) &\geq (s_1+s_2+s_3+s_4)2^m
\end{aligned}
$$

where $C_{\alpha\cdots\beta}$ means a set of constrained sequences having prefix α and postfix β.

Step 2

- The previous inequalities mainly concerns the numbers of code words.
- If (the left hand side) > (the right hand side), then we can discard code words in $C_{\alpha\cdots\beta}$, for example,

$$(s_1 + s_2 + s_3 + s_4)|C_{1\cdots 1}| \quad > \quad s_1 2^m$$

- If the frequency of 000 is small then the probability of the reset changes may be small.
- We discard sequences containing many consecutive zeros, e.g., $0001, 1000, 000101000$.
- We accept a code with low code rate and then get the freedom of adjustment of the resulting code.
- This adjustment or tuning is possible with almost no cost thanks to the Immink coding. So we can construct and try many different codes easily.

Step 2

- The previous inequalities mainly concerns the numbers of code words.
- If (the left hand side) $>$ (the right hand side), then we can discard code words in $C_{\alpha\cdots\beta}$, for example,

$$(s_1 + s_2 + s_3 + s_4)|C_{1\cdots1}| \quad > \quad s_1 2^m$$

- If the frequency of 000 is small then the probability of the reset changes may be small.
- We discard sequences containing many consecutive zeros, e.g., $0001, 1000, 000101000$.
- We accept a code with low code rate and then get the freedom of adjustment of the resulting code.
- This adjustment or tuning is possible with almost no cost thanks to the Immink coding. So we can construct and try many different codes easily.

Step 2

- The previous inequalities mainly concerns the numbers of code words.
- If (the left hand side) $>$ (the right hand side), then we can discard code words in $C_{\alpha \cdots \beta}$, for example,

$$(s_1 + s_2 + s_3 + s_4)|C_{1 \cdots 1}| \quad > \quad s_1 2^m$$

- If the frequency of 000 is small then the probability of the reset changes may be small.
- We discard sequences containing many consecutive zeros, e.g., $0001, 1000, 000101000$.
- We accept a code with low code rate and then get the freedom of adjustment of the resulting code.
- This adjustment or tuning is possible with almost no cost thanks to the Immink coding. So we can construct and try many different codes easily.

Step 2

- The previous inequalities mainly concerns the numbers of code words.
- If (the left hand side) > (the right hand side), then we can discard code words in $C_{\alpha\cdots\beta}$, for example,

$$(s_1 + s_2 + s_3 + s_4)|C_{1\cdots1}| \quad > \quad s_1 2^m$$

- If the frequency of 000 is small then the probability of the reset changes may be small.
- We discard sequences containing many consecutive zeros, e.g., $0001, 1000, 000101000$.
- We accept a code with low code rate and then get the freedom of adjustment of the resulting code.
- This adjustment or tuning is possible with almost no cost thanks to the Immink coding. So we can construct and try many different codes easily.

Step 2

- The previous inequalities mainly concerns the numbers of code words.
- If (the left hand side) $>$ (the right hand side), then we can discard code words in $C_{\alpha\cdots\beta}$, for example,

$$(s_1 + s_2 + s_3 + s_4)|C_{1\cdots 1}| \quad > \quad s_1 2^m$$

- If the frequency of 000 is small then the probability of the reset changes may be small.
- We discard sequences containing many consecutive zeros, e.g., $0001, 1000, 000101000$.
- We accept a code with low code rate and then get the freedom of adjustment of the resulting code.
- This adjustment or tuning is possible with almost no cost thanks to the Immink coding. So we can construct and try many different codes easily.

Step 2

- The previous inequalities mainly concerns the numbers of code words.
- If (the left hand side) $>$ (the right hand side), then we can discard code words in $C_{\alpha\cdots\beta}$, for example,

$$(s_1 + s_2 + s_3 + s_4)|C_{1\cdots1}| \quad > \quad s_1 2^m$$

- If the frequency of 000 is small then the probability of the reset changes may be small.
- We discard sequences containing many consecutive zeros, e.g., $0001, 1000, 000101000$.
- We accept a code with low code rate and then get the freedom of adjustment of the resulting code.
- This adjustment or tuning is possible with almost no cost thanks to the Immink coding. So we can construct and try many different codes easily.

Result 1 by Li Bojun[2017]

N: the total number of cells

$N_{1\Rightarrow 0}$: the number of reset changes

P_{reset}: the frequency of reset changes

$$P_{reset} = \frac{N_{1\Rightarrow 0}}{N}$$

Table: P_{reset}

Length	Cells reset	P_{reset}
90	20535	0.2281
900	209592	0.2329
1800	419471	0.2330

where 'length' means the length of blocks of encoded cells and random data were written on the cells 1000 times.

Result 2 by Li Bojun[2017]

	No coded	K. Cai	K. Cai	Proposed
k	-	$k=1$	$k=3$	$k=3$
Capacity	1	0.6942	0.9468	0.94468
Code length	-	13	18	9
Code rate	1	0.6923	0.9444	0.8889
P_{reset}	0.5	0.2773	0.4337	0.2230
R	0%	44.54%	13.26%	53.40%

where R is a rate of decrease in the frequency of reset changes and is

$$R = \frac{(P_{reset} \text{ of No Coded}) - P_{reset}}{P_{reset} \text{ of No Coded}} = \frac{0.5 - P_{reset}}{0.5}$$

- R of our code is the best among the above codes.
- The code rate of our code is lower than that of Cai's $k=3$ code but our code is better than Cai's $k=1$ code.
- Our code reduces the frequency of reset changes with relatively high code rate.

Result 2 by Li Bojun[2017]

	No coded	K. Cai	K. Cai	Proposed
k	-	$k = 1$	$k = 3$	$k = 3$
Capacity	1	0.6942	0.9468	0.94468
Code length	-	13	18	9
Code rate	1	0.6923	0.9444	0.8889
P_{reset}	0.5	0.2773	0.4337	0.2230
R	0%	44.54%	13.26%	53.40%

where R is a rate of decrease in the frequency of reset changes and is

$$R = \frac{(P_{reset} \text{ of No Coded}) - P_{reset}}{P_{reset} \text{ of No Coded}} = \frac{0.5 - P_{reset}}{0.5}$$

- R of our code is the best among the above codes.
- The code rate of our code is lower than that of Cai's $k = 3$ code but our code is better than Cai's $k = 1$ code.
- Our code reduces the frequency of reset changes with relatively high code rate.

Result 2 by Li Bojun[2017]

	No coded	K. Cai	K. Cai	Proposed
k	-	$k=1$	$k=3$	$k=3$
Capacity	1	0.6942	0.9468	0.94468
Code length	-	13	18	9
Code rate	1	0.6923	0.9444	0.8889
P_{reset}	0.5	0.2773	0.4337	0.2230
R	0%	44.54%	13.26%	53.40%

where R is a rate of decrease in the frequency of reset changes and is

$$R = \frac{(P_{reset} \text{ of No Coded}) - P_{reset}}{P_{reset} \text{ of No Coded}} = \frac{0.5 - P_{reset}}{0.5}$$

- R of our code is the best among the above codes.
- The code rate of our code is lower than that of Cai's $k=3$ code but our code is better than Cai's $k=1$ code.
- Our code reduces the frequency of reset changes with relatively high code rate.

Result 2 by Li Bojun[2017]

	No coded	K. Cai	K. Cai	Proposed
k	-	$k=1$	$k=3$	$k=3$
Capacity	1	0.6942	0.9468	0.94468
Code length	-	13	18	9
Code rate	1	0.6923	0.9444	0.8889
P_{reset}	0.5	0.2773	0.4337	0.2230
R	0%	44.54%	13.26%	53.40%

where R is a rate of decrease in the frequency of reset changes and is

$$R = \frac{(P_{reset} \text{ of No Coded}) - P_{reset}}{P_{reset} \text{ of No Coded}} = \frac{0.5 - P_{reset}}{0.5}$$

- R of our code is the best among the above codes.
- The code rate of our code is lower than that of Cai's $k=3$ code but our code is better than Cai's $k=1$ code.
- Our code reduces the frequency of reset changes with relatively high code rate.

New PCM

- There are new PCM technologies with new materials and with a new physical phenomenon, for example, interfacial phase change memory (IPCM).
- IPCM uses only the crystalline state and a laser beam changes its phase (resistance). There is no thermal interference among cells in IPCM.
- We must find a new constraint for IPCM.

Simpson, R.E.; P. Fons; A. V. Kolobov; T. Fukaya; et al. "Interfacial phase-change memory". Nature Nanotechnology, pp 501–505, July, 2011.

Conclusion

- Thermal interference of PCM.
- k-constraint for PCM.
- Immink coding for PCM.
- If we use the Immink coding, it is easy to tune or adjust the resulting code so that it has a good performance for PCM.

Reference

[1] Y.B.Liao，J.T.Lin，M.H.Chiang，and W.C.Hsu，"Assessment of novel phase change memory programming techniques"，Electron Devices and Slid-state Circuits，2008.EDSSC 2008.IEEE Tnternational Conference on，pp.1-4，Dec.2008

[2] Kui Cai,"Vertical Constrained Coding for Phase-Change Memory with Thermal Crosstalk," in Proc. IEEE International Conference on Computing, Networking and Communications, Invited Position Papers, pp. 312-316, Feb. 2014

[3] K.A.S.Immink, " Codes for mass data storage systems second edition ", Shannon Foundation Publishers, Eindhoven, The Netherlands, 2004.

[4] 李泊君，鎌部浩，路サン，"相変化メモリの熱クロストークを低減するための制約符号の構成"SITA2016，高山，岐阜，日本，2016.12.13-16.