Construction of RIO codes for improved SSD random read

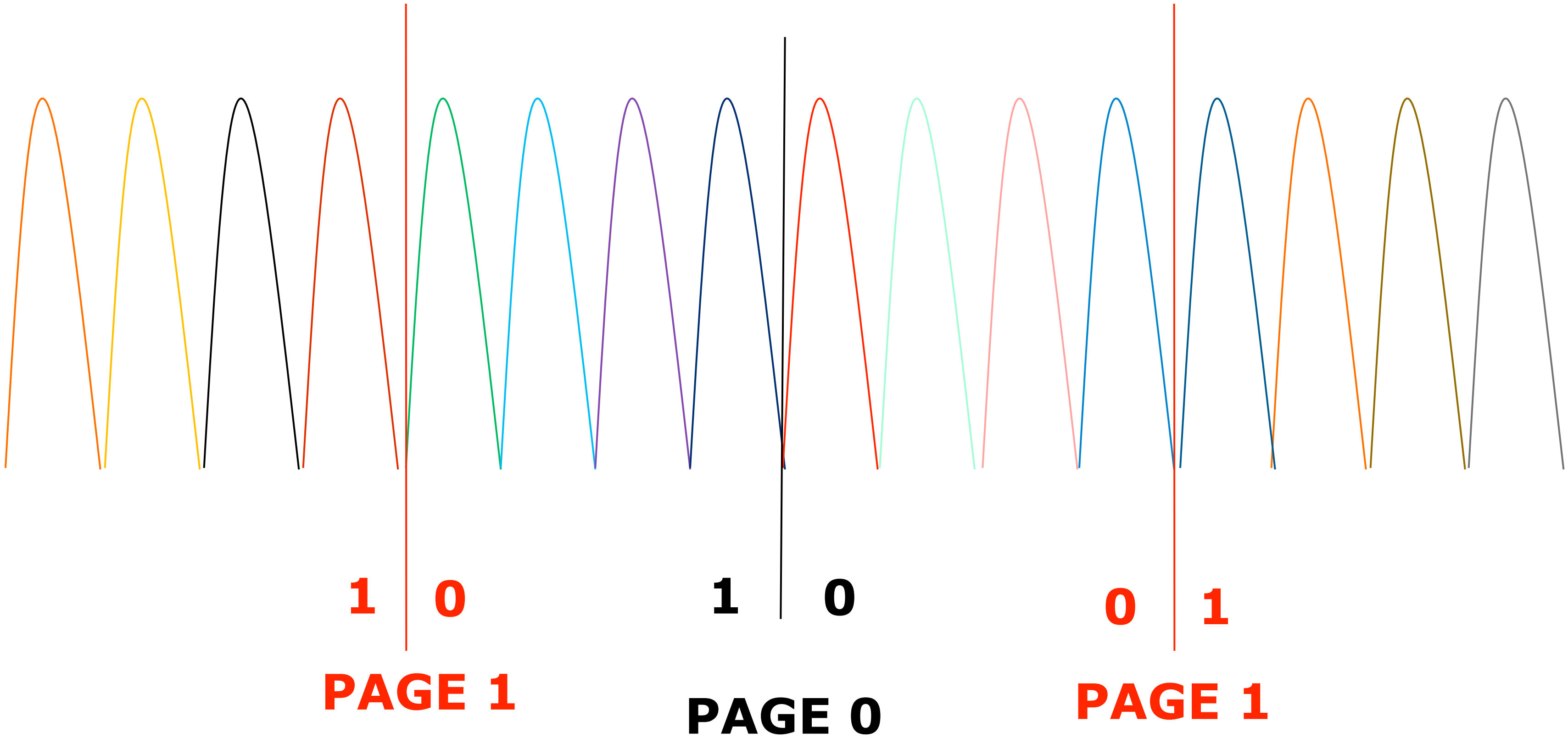
Ravi Motwani, Chong Ong
Intel Corporation

Thursday, August 15, 13

Random I/O Codes (RIO Codes)

- Introduced by Eran and Idan at NVM Workshop 2013
- Established a correspondence between WOM codes and RIO codes
- A WOM code which allows writing k bits into n cells t times
- RIO code which allows programming k bits into n cells for t pages such that each page can be read with a single strobe operation for the $t+1$ level NAND

16 level NAND- 4 Pages or 4 bpc

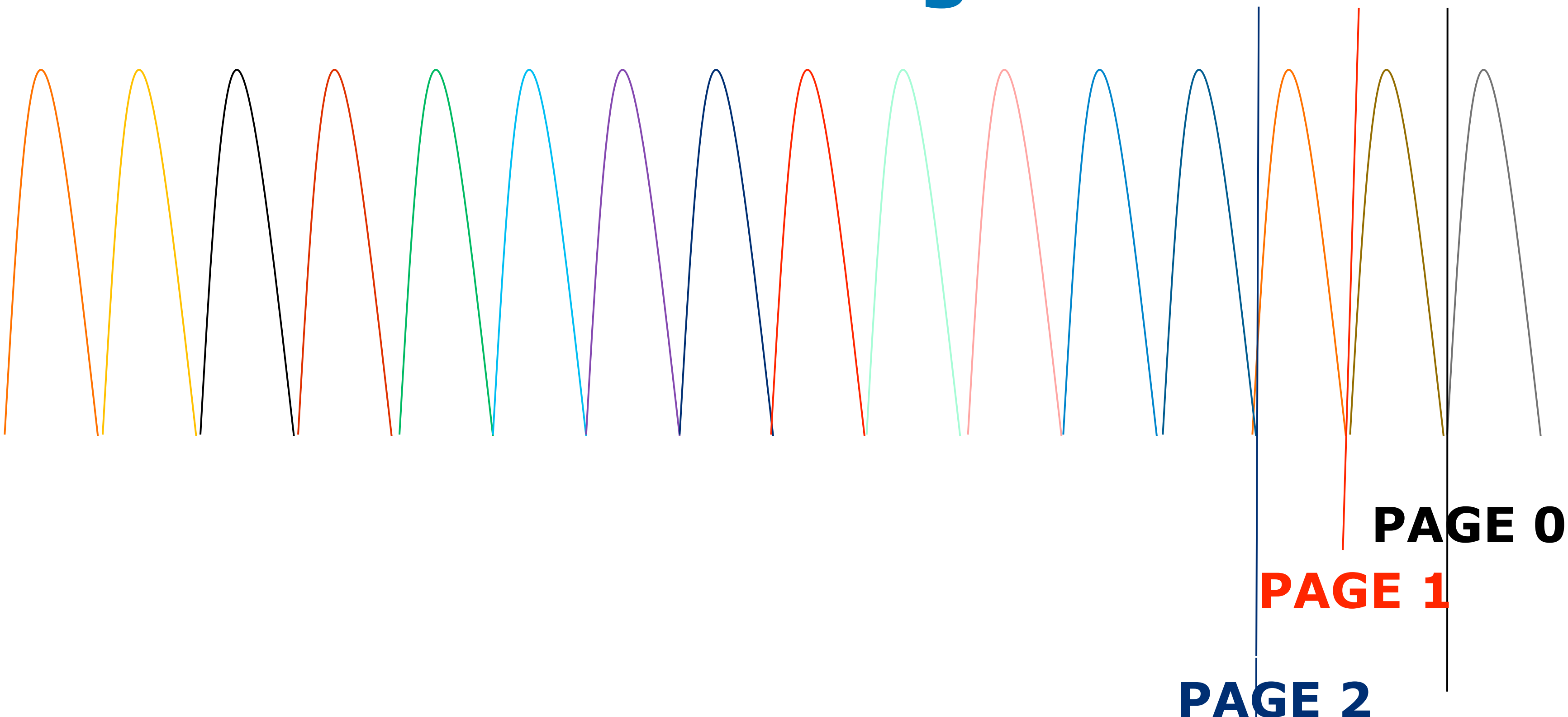


Need 1 sensing from NAND for Page 0, 2 senses for Page 1
4 senses for Page 2, 8 senses for Page 3

RIO Codes

- Strobes for 4 bpc
  - Page 0- 1 strobe
  - Page 1- 2 strobes
  - Page 2- 4 strobes
  - Page 3- 8 strobes
- For 4 bits per cell NAND, average reads to read one page is 3.75
- Read latency increases as levels increase
- To improve performance for MLC NAND, use Random I/O Codes

16 level NAND- 4 Pages



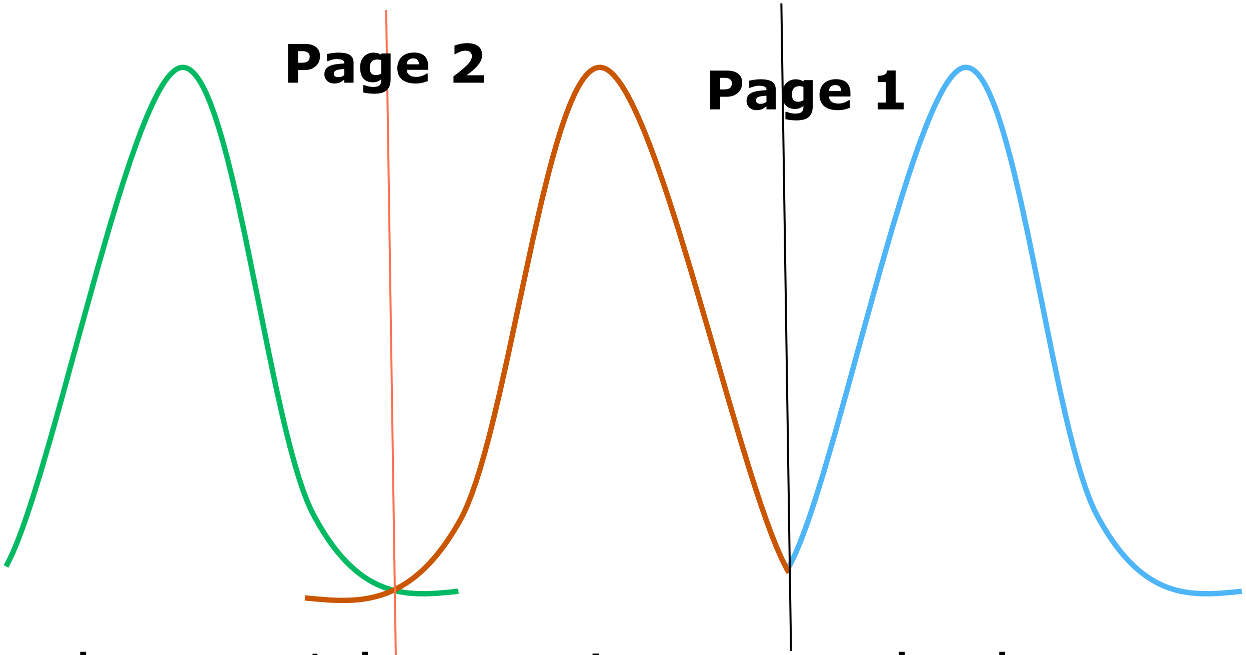
Every page can be read with a single sense
15 pages, redundancy across cells
Upper Limit- 4 bpc

WOM code

- Allows writing k bits to n cells t times without erasing (overwriting $t-1$ times)
- Example $(n,k,t) = (3,2,2)$ WOM code (1.33 bpc)

2 bits to write	1st write in SLC NAND	2nd write in SLC NAND
00	000	111
01	001	110
10	010	101
11	100	011

Three-level NAND RIO Code



- Read page 1 by putting a strobe between level 1 and level 2
- Read page 2 by putting a strobe between level 0 and level 1

Scope of this work

- For a given $n, k, t,$ does there exist an RIO code?
- If there exists a WOM code for given (n, k, t), then there exists an RIO code
- No existence proof for WOM codes
- There is a bound on the rate for RIO codes derived by Eran et. al.
- Even if the considered rate is lower than the bound there may not exist an RIO code
- An algorithm to check if there exists an RIO code for a given (n, k, t)

RIO Code (n, k, t)

- $n = 4, 2k = 6, t = 2$
- $k = 3, 3$ bits per page
- $2^6 = 64$ combinations to be programmed
- 4 cells can be programmed to $3^4 = 81$ possible states
- $81 - 64 = 17$ programming states have to be eliminated
- Two pages to be programmed
  - Page 1 read by putting a strobe between level 1 and level 2
  - Page 2 read by strobe between level 0 and level 1
- 8 combinations for each page (3 bits per page)
- Arrange the 8 combinations for each page as rows and columns of a 2-D matrix

Table with entries for the two pages

Bits from Page 2/ Page 1 read	1111 $yyyy$ $y - \{0,1\}$ A_{15}	1110 $yyy2$ A_{14}	1101 $yy2y$ A_{13}	1011 $y2yy$ A_{11}	0111 $2yyy$ A_7	$yy22$	$y2y2$	$y22y$
0000 $xxxx$ $x - \{1,2\}$ B_{15}	1111	1112	1121	1211	2111	?	?	?
0001 $xxx0$ B_{14}	1110	X	1120	1210	2110	?	?	?
0010 $xx0x$ B_{13}	1101	1102	X	1201	2101	?	?	?
0100 $x0xx$ B_{11}	1011	1012	1021	X	2011	?	?	?
1000 $0xxx$ B_7	0111	0112	0121	0211	X	?	?	?
$x00x$	?	0002	?	?	?	?	?	?
$00xx$	?	0012	?	?	?	?	?	?
$0x0x$	?	0102	?	?	?	?	?	?

RIO Code (n, k, t)

- 4 cells imply that there are 16 possibilities for bits read
- Combine 16 possibilities read for each page into 8 groups
- Each group is either a column (for Page 1) or a row (for Page 2)
- Any one possible read combination can occupy atmost one row or column

Enumerate Program states for Page 1

- Strobe between level 1 and level 2
- Read bits and programmed levels
- At most 8 programmed levels from each row can be used
- From the *yyyy* programmed levels, 8 entries cannot be used

Bits read from NAND	Programmed levels	Total number	Name of the set which stores the programmed levels
0000	2222	1	A_0
0001	$222y \quad y - \{0,1\}$	2	A_1
0010	22*y*2	2	A_2
0011	22*yy*	4	A_3
0100	2*y*22	2	A_4
0101	2*y*2*y*	4	A_5
0110	2*yy*2	4	A_6
0111	2*yyy*	8	A_7
1000	*y*222	2	A_8
1001	*y*22*y*	4	A_9
1010	*y*2*y*2	4	A_{10}
1011	*y*2*yy*	8	A_{11}
1100	*yy*22	4	A_{12}
1101	*yy*2*y*	8	A_{13}
1110	*yyy*2	8	A_{14}
1111	*yyyy*	16	A_{15}

Page 2

- Strobe between level 0 and level 1
- At most 8 programmed levels from each row can be used
- From the $xxxx$ programmed levels, 8 entries cannot be used

Bits read from NAND	Programmed levels	Total number	Name of the set which stores the programmed levels
0000	$xxxx \quad x-\{1,2\}$	16	B_{15}
0001	$xxx0$	8	B_{14}
0010	$xx0x$	8	B_{13}
0011	$xx00$	4	B_{12}
0100	$x0xx$	8	B_{11}
0101	$x0x0$	4	B_{10}
0110	$x00x$	4	B_9
0111	$x000$	2	B_8
1000	$0xxx$	8	B_7
1001	$0xx0$	4	B_6
1010	$0x0x$	4	B_5
1011	$0x00$	2	B_4
1100	$00xx$	4	B_3
1101	$00x0$	2	B_2
1110	$000x$	2	B_1
1111	0000	1	B_0

How many programming levels cannot be used

- Allowed to drop $3^4 - 2^6 = 17$ possibilities

- 8 programming states corresponding to set A_{15} cannot be used

- 8 programming states corresponding to set B_{15} cannot be used

- 16 programing states are out

How many programming states are not allowed?

Bits from Page 2/ Page 1 read	1111 $yyyy$ $y-\{0,1\}$ A_{15}	1110 $yyy2$ A_{14}	1101 $yy2y$ A_{13}	1011 $y2yy$ A_{11}	0111 $2yyy$ A_7
0000 $xxxx$ $x-\{1,2\}$ B_{15}	1111	1112	1121	1211	2111
0001 $xxx0$ B_{14}	1110	x	1120	1210	2110
0010 $xx0x$ B_{13}	1101	1102	x	1201	2101
0100 $x0xx$ B_{11}	1011	1012	1021	x	2011
1000 $0xxx$ B_7	0111	0112	0121	0211	x

How many programming states are out?

- In addition to the 16 already ousted, there are at least 4 more states which are not allowed
- 20 programming levels are forbidden by the RIO code
- Conclusion- There does not exist an RIO code for (4,3,2) case

Algorithm to check existence

- Generate intuition from $(4,3,2)$ case to generalize to any (n, k, t)

Existence for $(n, k, 2)$

- Define sets A and B for Page 1 and Page 2 resp.
- A_i^J defined for Page 1 such that it includes all the possible programmed n-cells such that
  - i out of n cells are programmed to level 0 or 1
  - J is the set of indices which are programmed to level 0 or 1
- Example
  - $n = 10, i = 6,\ J = \{1,2,3,7,8,10\}$
  - $A_6^J = \{0002220020, 0002220021, 0002220120, \ldots .1112221121\}$
- A_i^J has 2^i elements and for each i, total number of J sets are ${}^nC_i$
- For Page 2, define sets B_i^J where i out of n cells are programmed to level 1 or 2

Existence for $(n, k, 2)$

- Sets A_i^J and $B_i^J, i = 0,1, \ldots .n$
- A_n^J and B_n^J each have 2^n elements each
- Upto 2^k elements can be chosen from each of these sets
- $2(2^n$ - $2^k)$ programmed states are out
- Sets A_{n-1}^J and B_{n-1}^J each have 2^{n-1} elements
- Upto 2^k elements can be chosen from each set
- $2n(2^{n-1}$-$2^k)$programmed states are out

Existence for $(n, k, 2)$

- For set A_i^J, out of the 2^{n-i} programmed states, 2^k can only be taken
- Total programmed states dropped out for all A_i^J and B_i^J are $2.{}^nC_i.(2^{n-i} - 2^k)$
- Sum the drop-outs over all i to get

$$\sum_{i=0}^{n} 2.{}^nC_i.(2^{n-i} - 2^k),\ i \ni n - i > k$$

Drop-outs due to intersection of sets A and B

- Intersection of programming states from sets A_i^J and B_l^Y
- If sets J^c and Y^c intersect, the elements in the intersection cannot be programmed
- Those are the drop-outs
- A_n^J and B_n^Y lead to no drop-outs since $J^c \cap Y^c = \emptyset$
- A_{n-1}^J and B_{n-1}^Y have one drop-out as Y runs across the n combinations for a fixed J
- n dropouts follow from the intersection of A_{n-1}^J and B_{n-1}^Y as we span all sets J and Y

Drop-outs due to intersection of states from A and B

- A_{n-1}^{J} and B_{n-2}^{Y} have at least one drop-out as Y runs across the n combinations for a fixed J
- Given the locations programmed to level 2 in set J^c, the probability that location intersects with a location in Y^c for Y in B_{n-2}^{Y} equals $\frac{2}{n}$
- Total possibilities for Y in B_{n-2}^{Y} is ${}^{n}C_2$
- Total drop-outs $= {}^{n}C_2 \, \frac{2}{n}$
- Total drop-outs over all J $= n \, {}^{n}C_2 \, \frac{2}{n}$

Drop-outs due to intersection of states from A and B

- Intersection of states in A_i^J and B_l^Y, $i \leq l$
- Consider A_i^J, what is the probability that the l coordinates in Y^c match with any one of the i coordinates in J^c?
- Probability that match at none of the i locations =

$$P = \frac{n-l}{n} \cdot \frac{n-l-1}{n-1} \cdot \frac{n-l-2}{n-2} \cdots \frac{n-(l-i+1)}{n-(i-1)}$$

- Given the locations programmed to level 2 in set J^c, the probability that location intersects with any of the location in Y^c of B_l^Y equals $1 - P$
- Total drop-outs = ${}^nC_l\,{}^nC_i\,(1 - P)$

Construction by filling table

	A_n^J	$A_{n-1}^{J_1}$	$A_{n-1}^{J_2}$	$A_{n-1}^{J_3}$	.....	$A_{n-1}^{J_n}$	$A_{n-2}^{J_{n+1}}$	$A_{n-2}^{J_{n+2}}$	...
B_n^Y	$J^c \cap Y^c = \emptyset$								
$B_{n-1}^{Y_1}$									
$B_{n-1}^{Y_2}$									
$B_{n-1}^{Y_3}$									
$B_{n-1}^{Y_n}$									
$B_{n-2}^{Y_{n+1}}$									
$B_{n-2}^{Y_{n+2}}$									

Generalized RIO Coding

- RIO code does not exist for (4,3,2)
- If requirement is 1.5 bpc
- Generalized RIO Code- Permits reading some pages with one read and others allowed to have 2 or multiple reads

Generalized RIO code with 1.5 bpc for t=2

- 4 cells store 6 bits -> 1.5bpc
- 2 pages with 3 bits each

	Strobe L1 | L2 L0 | L1	1111	1110	1101	1011	0111	1100/ 1010	1001/ 0110 1000	010/10 100/ 0011
	UP/LP	000	001	010	011	100	101	110	111
1110/0001	000	0001	0002	1120	1210	2110	0122	1220	2210
1101/0010	001	0010	1102	0020	1201	2101	1202	2102	2201
1011/0100	010	0100	1012	1021	0200	2011	1022	2012	2021
1001/0110	011	0110	1002	0120	0210	2001	1122	2002	2121
0111/1000	100	1000	0112	0121	0211	2000	0212	0221	2122
0101/1010	101	1010	0102	1020	0201	2010	0202	1222	2020
0011/1100	110	1100	0012	0021	1200	2100	0022	1221	2200
0000/1111	111	0000	1112	1121	1211	2111	1212	2112	2211

Generalized RIO code for $(4,3,2)$

- The mapping permits reading one page (LP or Page 1) with a single strobe
- Applying a strobe between level 1 and level 2 gives 4 bits from 4 cells which can be uniquely mapped to 3 bits
- The second page (UP or Page 2) needs 2 strobes for unique decoding
- Page 2 read needs more than one strobe if the bits read out are 0000

Generalize for any t

- For $t = 3$, we get a cube and for $t = 4$ onwards, a hypercube
- Principal of checking for existence and construction remain the same

Mappings for minimizing RBER

- Once the mapping of the sets is done, it remains to decide how to map data bits to the RIO code bits
- For the $(4,3,2)$ code, we need to map the 3 user or data bits to the 4 bits read from the NAND
- Mapping done to minimize the RBER
- n-cell voltages are closer to each other implies that the corresponding data bits are also close in Hamming distance
- 0001 and 0002 are two programmed states which are closest, so the assigned data bits are 000 and 001 resp.
- Reduces the BER amplification

Impact of RIO coding on tI/O

- The transfer time increases due to redundancy of the RIO code
- (4,3,2) RIO code
- Instead of 3 bits output from the NAND (conventional), 4 bits are output (RIO code)
- Overhead of 33% as far as tI/O is concerned
- Solution- Perform RIO demapping within the NAND

Generating soft information on input bits

- 3 data bits mapped to 4 RIO code bits
- $[u_0\ u_1\ u_2]$ ---> $[c_0\ c_1\ c_2\ c_3]$
- Read from NAND $[r_0\ r_1\ r_2\ r_3]$
- $LLR(u_0) = \ln \frac{P(u_0=0 \mid [r_0\ r_1\ r_2\ r_3])}{P(u_0=1 \mid [r_0\ r_1\ r_2\ r_3])}$

$$= \ln \frac{P([r_0\ r_1\ r_2\ r_3] \mid u_0=0)}{P([r_0\ r_1\ r_2\ r_3] \mid u_0=1)}$$

$$= \ln \frac{\sum_{[c_0\ c_1\ c_2\ c_3] \ni u_0=0} P([r_0\ r_1\ r_2\ r_3] \mid [c_0\ c_1\ c_2\ c_3])}{\sum_{[c_0\ c_1\ c_2\ c_3] \ni u_0=1} P([r_0\ r_1\ r_2\ r_3] \mid [c_0\ c_1\ c_2\ c_3])}$$

$$= \ln \frac{\sum_{[c_0\ c_1\ c_2\ c_3] \ni u_0=0} P(r_0|c_0)P(r_1|c_1)P(r_2|c_2)P(r_3|c_3)}{\sum_{[c_0\ c_1\ c_2\ c_3] \ni u_0=1} P(r_0|c_0)P(r_1|c_1)P(r_2|c_2)P(r_3|c_3)}$$

Summary and Conclusions

- A method to find if an RIO code exists for given parameters (n, k, t)
- The method is also constructive, it yields an RIO code
- Generalized RIO Code
- Mapping of user/data bits to the NAND read bits
- Impact on tI/O
- Generation of soft information

Thursday, August 15, 13