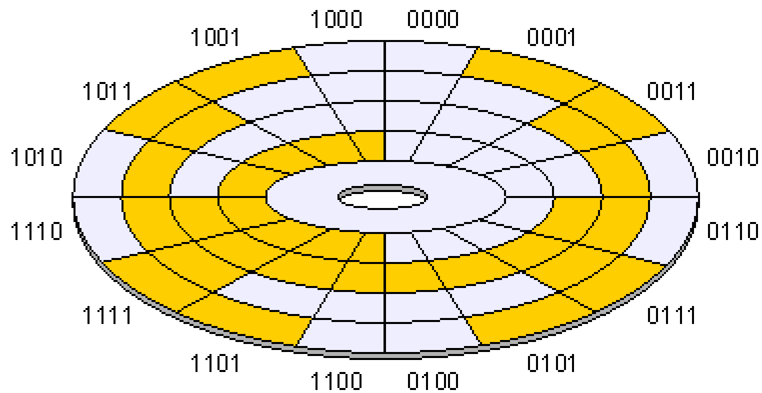
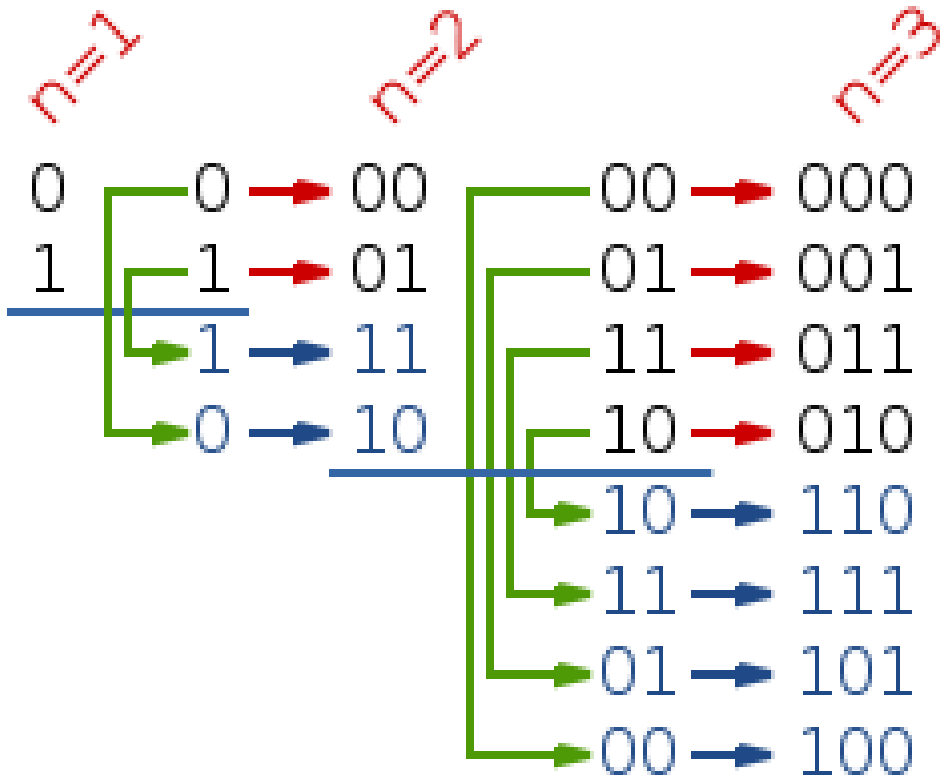
Digital Fundamentals

More on Gray Codes



- Gray numbers are unidistance numbers
  - unlike binary numbers only one bit changes from one count to another count.
- Gray pointers too have problem of metastability while synchronizing with other clock domains
  - but it is minimized by the fact of one bit change.
- Metastability condition on one bit causes +/- 1 count error
  - that is better compared to +/-8 count error in binary pointers.
- Because of this minimized error gray counters are generally used as FIFO pointers.

2^n: Power-of-2 Gray Code



Binary ⇔ Gray-Coding

Decimal Value	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

Conversion: Binary $\rightarrow$ Gray Code

- $B=\langle b_{n-1}, b_{n-2}, \ldots, b_1, b_0\rangle$: Binary Code
- $G=\langle g_{n-1}, g_{n-2}, \ldots, g_1, g_0\rangle$: Gray Code
- Gray code is a non-redundant representation.
- We have:
  - $g_{n-1} = b_{n-1}$
  - $g_i = b_{i+1} \oplus b_i$, $i=n-2, \ldots, 0$.
- Ex. $B=\langle 1, 1, 0, 1\rangle$

  $\Rightarrow G=\langle b_3, b_3 \oplus b_2, b_2 \oplus b_1, b_1 \oplus b_0\rangle$

  $\Rightarrow G=\langle 1, 0, 1, 1\rangle$

Conversion: Gray $\rightarrow$ Binary Code

- For the MSB,
  - $b_{n-1} = g_{n-1}$
- For i = n–2, …, 0,
  - $b_i = b_{i+1} \oplus g_i$

- Ex. G=<1, 1, 0, 1>

  $\Rightarrow$ $B = \langle g_3, g_3 \oplus g_2, g_3 \oplus g_2 \oplus g_1, g_3 \oplus g_2 \oplus g_1 \oplus g_0 \rangle$

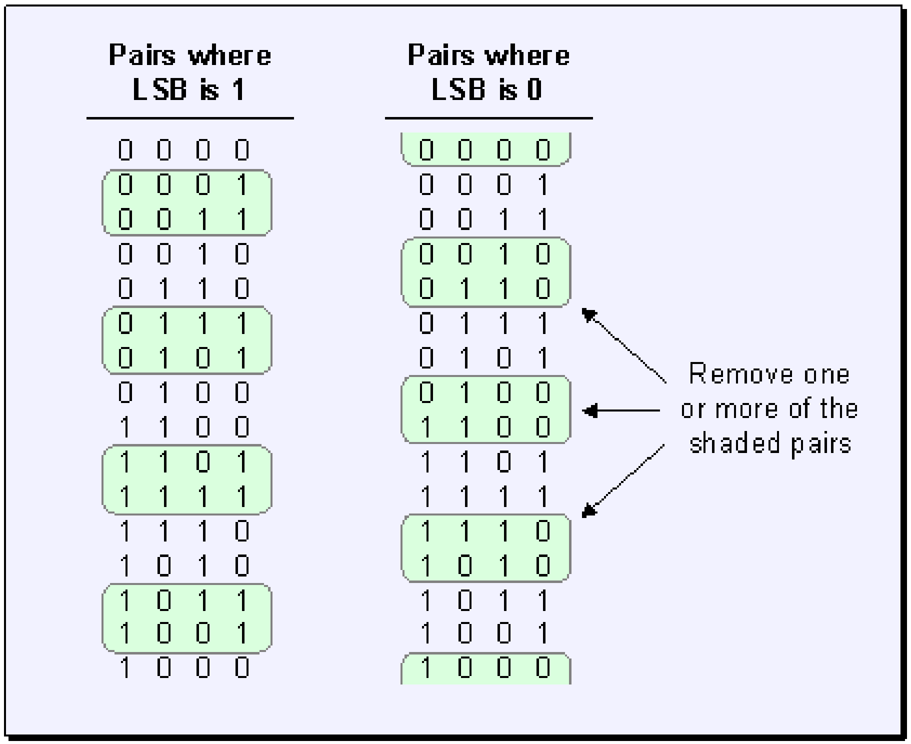
  $\Rightarrow$ B=<1, 0, **1?**, **1?**>

Gray Code of Any Length

- FIFO counters can be designed to have any mod number.
- FIFO memory locations can also be any arbitrary number.
- The dimension of an FIFO could be any length.

- Question:
  - Can we have a gray code of any length?

Even Length:
Throwing Away Adjacent Pairs with the same LSB



- Removing one pair
  - a 14-count sequence
- removing any two pairs
  - a 12-count sequence
- removing any three pairs
  - a 10-count sequence
- pointless to remove four pairs to give us an 8-count sequence,
  - as we could achieve the same effect by dropping down to a 3-bit Gray code).

Even Length: Pruning the Middle

16-Count	14-Count	12-Count	10-Count
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
0 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1
0 0 1 0	0 0 1 0	0 0 1 0	0 0 1 0
0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0
0 1 1 1	0 1 1 1	0 1 1 1	~~0 1 1 1~~
0 1 0 1	0 1 0 1	~~0 1 0 1~~	~~0 1 0 1~~
0 1 0 0	~~0 1 0 0~~	~~0 1 0 0~~	~~0 1 0 0~~
1 1 0 0	~~1 1 0 0~~	~~1 1 0 0~~	~~1 1 0 0~~
1 1 0 1	1 1 0 1	~~1 1 0 1~~	~~1 1 0 1~~
1 1 1 1	1 1 1 1	1 1 1 1	~~1 1 1 1~~
1 1 1 0	1 1 1 0	1 1 1 0	1 1 1 0
1 0 1 0	1 0 1 0	1 0 1 0	1 0 1 0
1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1
1 0 0 1	1 0 0 1	1 0 0 1	1 0 0 1
1 0 0 0	1 0 0 0	1 0 0 0	1 0 0 0

"Mirror" line

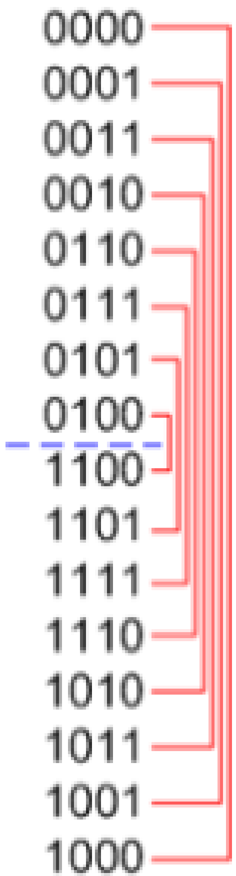
- The previous solution is easy to use, but it would require us to keep track as to which pairs of codes we've removed.
- The "mirroring method" has a more correct name:
  - a "recursive reverse-and-prefix" approach.
- Rule:
  - Simply remove pairs of entries from the center of the table around the "mirror line"

Even Length: Pruning the Ends

16-Count	14-Count	12-Count	10-Count	
0 0 0 0	~~0 0 0 0~~	~~0 0 0 0~~	~~0 0 0 0~~	
0 0 0 1	0 0 0 1	~~0 0 0 1~~	~~0 0 0 1~~	
0 0 1 1	0 0 1 1	0 0 1 1	~~0 0 1 1~~	
0 0 1 0	0 0 1 0	0 0 1 0	0 0 1 0	
0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	
0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	
0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1	
0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	"Mirror" line
1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	
1 1 0 1	1 1 0 1	1 1 0 1	1 1 0 1	
1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	
1 1 1 0	1 1 1 0	1 1 1 0	1 1 1 0	
1 0 1 0	1 0 1 0	1 0 1 0	1 0 1 0	
1 0 1 1	1 0 1 1	1 0 1 1	~~1 0 1 1~~	
1 0 0 1	1 0 0 1	~~1 0 0 1~~	~~1 0 0 1~~	
1 0 0 0	~~1 0 0 0~~	~~1 0 0 0~~	~~1 0 0 0~~	

- Remove the same numbers of entries from the top and from the bottom of a traditional power-of-2 Gray code counter.

No Gray Code of Odd Length



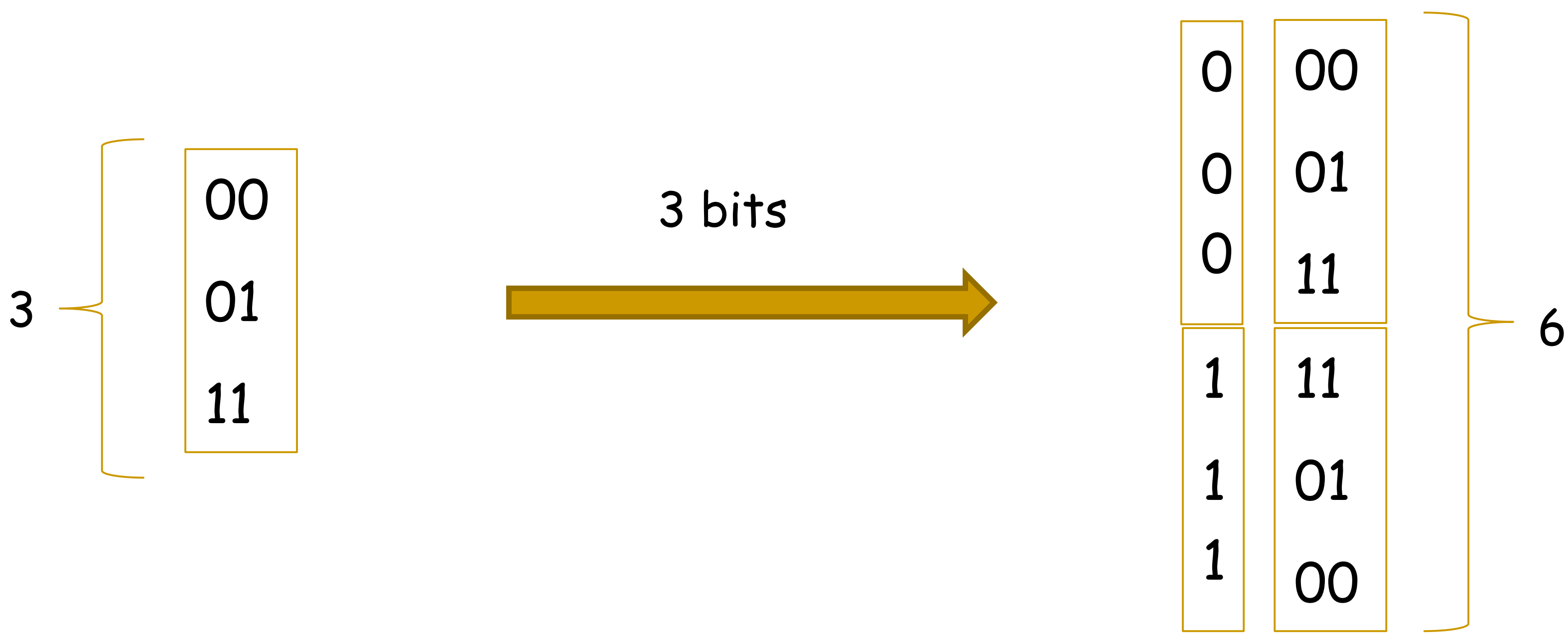
- Lemma:
  - It is not possible to construct a Gray code with an odd cycle length.
- Proof:
  - As only one bit changes at a time, the parity of the code toggles.
  - Parity(0000)=even, Parity(0001)=odd, Parity(0011)=even, ……
  - As we have an odd number of codes, it contradicts the Gray hypothesis of cyclic feature.

Gray Code Counter of Length 3 ?

3

00
01
11

A Virtual Space based Gray Code



Shannon's Decomposition

$$Y = f(x_1, x_2, \ldots, x_n) = x_i f_{x_i} + \bar{x}_i f_{\bar{x}_i}$$

where

$$f_{x_i=1} = f_{x_i} = f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

$$f_{x_i=0} = f_{\bar{x}_i} = f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$$

Proof of Shannon Expansion

- Proof of Shannon's expansion theorem
- $f(x_1, x_2, \ldots, x_n) = x_1{}' f(0, x_2, \ldots, x_n) + x_1\ f(1, x_2, \ldots, x_n)$
- This theorem can be proved using **perfect induction**, by showing that the expression is true for every possible value of x_1.
- As x_1 is a Boolean variable, we look at only two cases: $x_1 = 0$ and $x_1 = 1$.
- Setting $x_1 = 0$, we have:
  - $f(0, x_2, \ldots, x_n) = 1 \wedge f(0, x_2, \ldots, x_n) + 0 \wedge f(1, x_2, \ldots, x_n) = f(0, x_2, \ldots, x_n)$
- Setting $x_1 = 1$, we have:
  - $f(1, x_2, \ldots, x_n) = 0 \wedge f(0, x_2, \ldots, x_n) + 1 \wedge f(1, x_2, \ldots, x_n) = f(1, x_2, \ldots, x_n)$
- This proof can be performed for any arbitrary xi in the same manner.

References

- The Art of Hardware Architecture: Design Methods and Techniques for Digital Circuits by Mohit Arora. Springer.
- http://www.eetimes.com/document.asp?doc_id=1274581
- https://forums.xilinx.com/t5/BRAM-FIFO/FIFO-with-non-power-of-two-depth/td-p/482158
- http://www.verilogpro.com/asynchronous-fifo-design/
- Patent US7667629.