KODE HAMMING

Komunikasi Data

- Kode Hamming merupakan kode non-trivial untuk koreksi kesalahan yang pertama kali diperkenalkan.
- Kode ini dan variansinya telah lama digunakan untuk kontrol kesalahan pada sistem komunikasi digital.
- Kode Hamming biner dapat direpresentasikan dalam bentuk persamaan:

  $(n,k) = (2^m-1, 2^m-1-m)$

- Contoh:

  jika m = jumlah paritas = 3

  k = jumlah data = 4

  n = jumlah bit informasi yang membentuk n sandi = 7

  maka kode Hamming nya adalah C (7,4) dengan dmin =3

Forward Error Correction

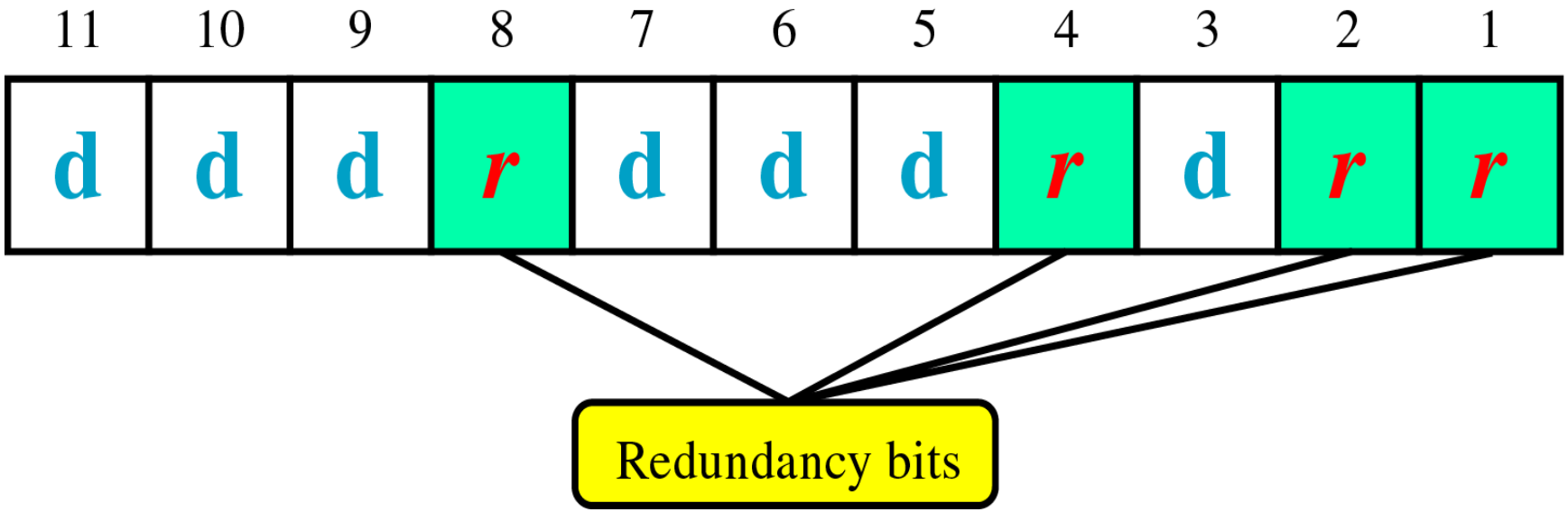
- Error Correcting codes dinyatakan sebagai penerusan koreksi kesalahan untuk mengindikasikan bahwa pesawat penerima sedang mengoreksi kesalahan.
- Kode pendeteksi yang paling banyak digunakan merupakan kode Hamming.
- Posisi bit-bit Hamming dinyatakan dalam 2^n dengan n bilangan bulat sehingga bit-bit Hamming akan berada dalam posisi 1, 2, 4, 8, 16, dst..

Error Correction(cont'd)

- Hamming Code

  ~ developed by R.W.Hamming

- positions of redundancy bits in Hamming code



The key to the Hamming Code is the use of extra parity bits to allow the identification of a single error. Create the code word as follows:

- Mark all bit positions that are powers of two as parity bits. (positions 1, 2, 4, 8, 16, 32, 64, etc.)
- All other bit positions are for the data to be encoded. (positions 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.)
- Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.

  Position 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (1,3,5,7,9,11,13,15,...)
  Position 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (2,3,6,7,10,11,14,15,...)
  Position 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. (4,5,6,7,12,13,14,15,20,21,22,23,...)
  Position 8: check 8 bits, skip 8 bits, check 8 bits, skip 8 bits, etc. (8-15,24-31,40-47,...)
  Position 16: check 16 bits, skip 16 bits, check 16 bits, skip 16 bits, etc. (16-31,48-63,80-95,...)
  Position 32: check 32 bits, skip 32 bits, check 32 bits, skip 32 bits, etc. (32-63,96-127,160-191,...)
  etc.

- Set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7	d8	d9	d10	d11	p5	d12	d13	d14	d15	
Parity bit coverage	p1	X		X		X		X		X		X		X		X		X		X		...
	p2		X	X			X	X			X	X			X	X			X	X		
	p3				X	X	X	X					X	X	X	X					X	
	p4								X	X	X	X	X	X	X	X						
	p5																X	X	X	X	X	

Example:

A byte of data: 10011010

Create the data word, leaving spaces for the parity bits: _ _ 1 _ 0 0 1 _ 1 0 1 0

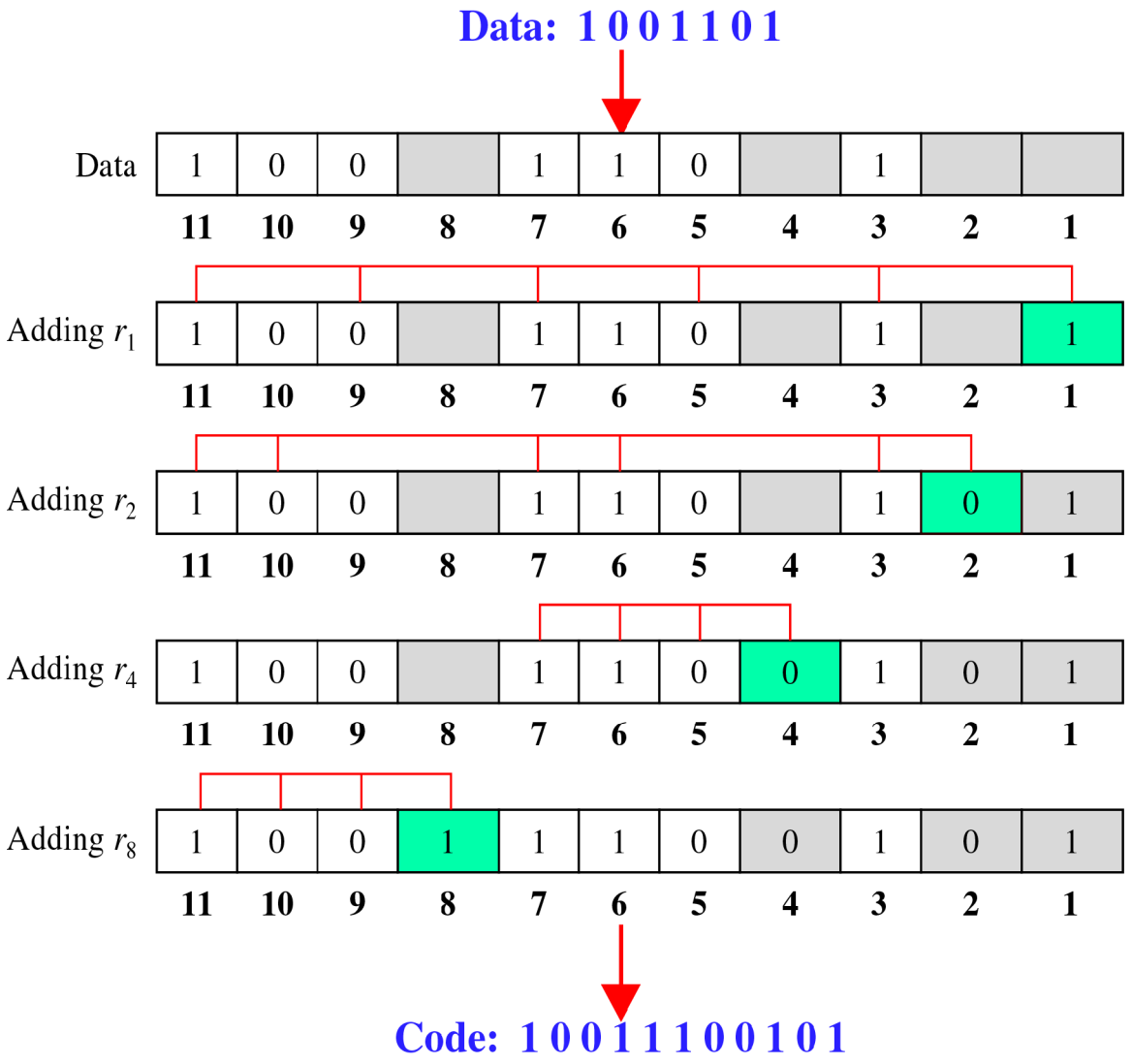
Calculate the parity for each parity bit (a ? represents the bit position being set):

- Position 1 checks bits 1,3,5,7,9,11:
  ? _ 1 _ 0 0 1 _ 1 0 1 0. Even parity so set position 1 to a 0: 0 _ 1 _ 0 0 1 _ 1 0 1 0
- Position 2 checks bits 2,3,6,7,10,11:
  0 ? 1 _ 0 0 1 _ 1 0 1 0. Odd parity so set position 2 to a 1: 0 1 1 _ 0 0 1 _ 1 0 1 0
- Position 4 checks bits 4,5,6,7,12:
  0 1 1 ? 0 0 1 _ 1 0 1 0. Odd parity so set position 4 to a 1: 0 1 1 1 0 0 1 _ 1 0 1 0
- Position 8 checks bits 8,9,10,11,12:
  0 1 1 1 0 0 1 ? 1 0 1 0. Even parity so set position 8 to a 0: 0 1 1 1 0 0 1 0 1 0 1 0

- Code word: 011100101010.

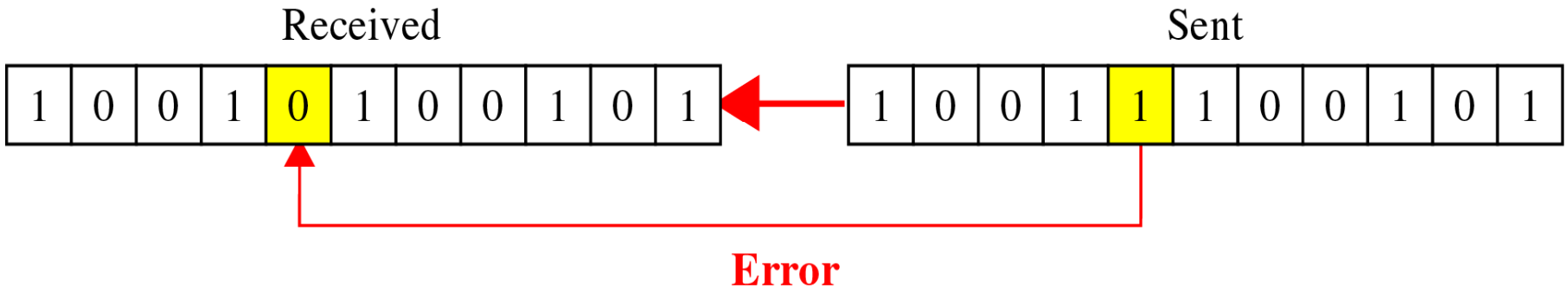
Error Correction(cont'd)

- Calculating the r values



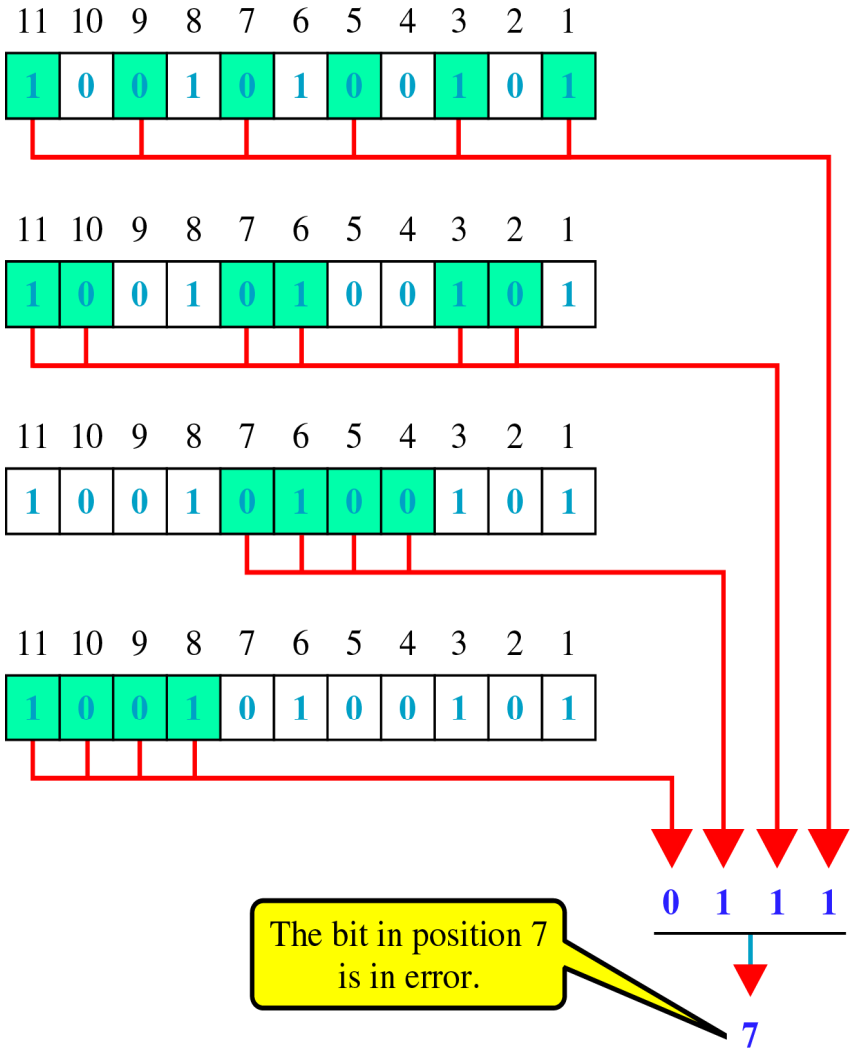
Contoh Error Correction(cont'd)

- Error Detection and Correction



Error Correction(cont'd)

- Error detection using Hamming Code



Error Correction(cont'd)

- Multiple-Bit Error Correction
  - redundancy bits calculated on overlapping sets of data units can also be used to correct multiple-bit errors.

  Ex) to correct double-bit errors, we must take into consideration that two bits can be a combination of any two bits in the entire sequence