WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Afrikaans
Català
Čeština
Dansk
Deutsch
Español
فارسی
Français
Gaeilge
한국어
Italiano
עברית
Magyar
Македонски
[illegible]
Nederlands
日本語
Norsk nynorsk
Polski
Português
Русский
Slovenčina
Српски / srpski
Svenska
ไทย
Türkçe
Українська
Tiếng Việt
中文
Edit links

Parity bit

From Wikipedia, the free encyclopedia

This article **needs additional citations for verification**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. *(January 2013)*

7 bits of data	(count of 1 bits)	8 bits including parity	
		even	odd
0000000	0	0000000**0**	0000000**1**
1010001	3	1010001**1**	1010001**0**
1101001	4	1101001**0**	1101001**1**
1111111	7	1111111**1**	1111111**0**

A **parity bit**, or **check bit** is a bit added to the end of a string of binary code that indicates whether the number of bits in the string with the value one is even or odd. Parity bits are used as the simplest form of error detecting code.

There are two variants of parity bits: **even parity bit** and **odd parity bit**.

In the case of even parity, the number of bits whose value is 1 in a given set are counted. If that total is odd, the parity bit value is set to 1, making the total count of 1's in the set an even number. If the count of ones in a given set of bits is already even, the parity bit's value remains 0.

In the case of odd parity, the situation is reversed. Instead, if the sum of bits with a value of 1 is odd, the parity bit's value is set to zero. And if the sum of bits with a value of 1 is even, the parity bit value is set to 1, making the total count of 1's in the set an odd number.

Even parity is a special case of a cyclic redundancy check (CRC), where the 1-bit CRC is generated by the polynomial x+1.

If the parity bit is present but not used, it may be referred to as **mark parity** (when the parity bit is always 1) or **space parity** (the bit is always 0).

Contents [hide]

Parity [edit]

In mathematics, parity refers to the evenness or oddness of an integer, which for a binary number is determined only by the least significant bit. In telecommunications and computing, parity refers to the evenness or oddness of the number of bits with value one within a given set of bits, and is thus determined by the value of all the bits. It can be calculated via an XOR sum of the bits, yielding 0 for even parity and 1 for odd parity. This property of being dependent upon all the bits and changing value if any one bit changes allows for its use in error detection schemes.

Error detection [edit]

If an odd number of bits (including the parity bit) are transmitted incorrectly, the parity bit will be incorrect, thus indicating that a **parity error** occurred in the transmission. The parity bit is only suitable for detecting errors; it cannot correct any errors, as there is no way to determine which particular bit is corrupted. The data must be discarded entirely, and re-transmitted from scratch. On a noisy transmission medium, successful transmission can therefore take a long time, or even never occur. However, parity has the advantage that it uses only a single bit and requires only a number of XOR gates to generate. See Hamming code for an example of an error-correcting code.

Parity bit checking is used occasionally for transmitting ASCII characters, which have 7 bits, leaving the 8th bit as a parity bit.

For example, the parity bit can be computed as follows, assuming we are sending simple 4-bit values 1001.

Type of bit parity	Successful transmission scenario
Even parity	A wants to transmit: 1001 A computes parity bit value: 1+0+0+1 (mod 2) = 0 A adds parity bit and sends: 10010 B receives: 10010 B computes parity: 1+0+0+1+0 (mod 2) = 0 B reports correct transmission after observing expected even result.
Odd parity	A wants to transmit: 1001 A computes parity bit value: 1+0+0+1 + **1** (mod 2) = 1 A adds parity bit and sends: 10011 B receives: 10011 B computes overall parity: 1+0+0+1+1 (mod 2) = 1 B reports correct transmission after observing expected result.

This mechanism enables the detection of single bit errors, because if one bit gets flipped due to line noise, there will be an incorrect number of ones in the received data. In the two examples above, B's calculated parity value matches the parity bit in its received value, indicating there are no single bit errors. Consider the following example with a transmission error in the second bit using XOR:

Type of bit parity error	Failed transmission scenario
Even parity Error in the second bit	A wants to transmit: 1001 A computes parity bit value: 1^0^0^1 = 0 A adds parity bit and sends: 10010 **...TRANSMISSION ERROR...** B receives: 1**1**010 B computes overall parity: 1^1^0^1^0 = 1 B reports incorrect transmission after observing unexpected odd result.
Even parity Error in the parity bit	A wants to transmit: 1001 A computes even parity value: 1^0^0^1 = 0 A sends: 10010 **...TRANSMISSION ERROR...** B receives: 1001**1** B computes overall parity: 1^0^0^1^1 = 1 B reports incorrect transmission after observing unexpected odd result.

There is a limitation to parity schemes. A parity bit is only guaranteed to detect an odd number of bit errors. If an even number of bits have errors, the parity bit records the correct number of ones, even though the data is corrupt. (See also error detection and correction.) Consider the same example as before with an even number of corrupted bits:

Type of bit parity error	Failed transmission scenario
	A wants to transmit: 1001 A computes even parity value: 1^0^0^1 = 0

Even parity Two corrupted bits	A sends: 10010 **...TRANSMISSION ERROR...** B receives: 1**1**01**1** B computes overall parity: 1^1^0^1^1 = 0 B reports correct transmission though actually incorrect.

B observes even parity, as expected, thereby failing to catch the two bit errors.

Usage [edit]

Because of its simplicity, parity is used in many hardware applications where an operation can be repeated in case of difficulty, or where simply detecting the error is helpful. For example, the SCSI and PCI buses use parity to detect transmission errors, and many microprocessor instruction caches include parity protection. Because the I-cache data is just a copy of main memory, it can be disregarded and re-fetched if it is found to be corrupted.

In serial data transmission, a common format is 7 data bits, an even parity bit, and one or two stop bits. This format neatly accommodates all the 7-bit ASCII characters in a convenient 8-bit byte. Other formats are possible; 8 bits of data plus a parity bit can convey all 8-bit byte values.

In serial communication contexts, parity is usually generated and checked by interface hardware (e.g., a UART) and, on reception, the result made available to the CPU (and so to, for instance, the operating system) via a status bit in a hardware register in the interface hardware. Recovery from the error is usually done by retransmitting the data, the details of which are usually handled by software (e.g., the operating system I/O routines).

RAID [edit]

Parity data is used by some RAID levels to achieve redundancy. If a drive in the array fails, remaining data on the other drives can be combined with the parity data (using the Boolean XOR function) to reconstruct the missing data.

For example, suppose two drives in a three-drive RAID 5 array contained the following data:

Drive 1: **01101101**
Drive 2: **11010100**

To calculate parity data for the two drives, an XOR is performed on their data:

```
    01101101
XOR 11010100
____________
    10111001
```

The resulting parity data, **10111001**, is then stored on Drive 3.

Should any of the three drives fail, the contents of the failed drive can be reconstructed on a replacement drive by subjecting the data from the remaining drives to the same XOR operation. If Drive 2 were to fail, its data could be rebuilt using the XOR results of the contents of the two remaining drives, Drive 1 and Drive 3:

Drive 1: **01101101**
Drive 3: **10111001**

as follows:

```
    10111001
XOR 01101101
____________
    11010100
```

The result of that XOR calculation yields Drive 2's contents. **11010100** is then stored on Drive 2, fully repairing the array. This same XOR concept applies similarly to larger arrays, using any number of disks. In the case of a RAID 3 array of 12 drives, 11 drives participate in the XOR calculation shown above and yield a value that is then stored on the dedicated parity drive.

History [edit]

A "parity track" was present on the first magnetic tape data storage in 1951. Parity in this form, applied across multiple parallel signals, is known as a transverse redundancy check. This can be combined with parity computed over multiple bits sent on a single signal, a longitudinal redundancy check. In a parallel bus, there is one longitudinal redundancy check bit per parallel signal.

Parity was also used on at least some paper-tape (punched tape) data entry systems (which preceded magnetic tape systems). On the systems sold by British company ICL (formerly ICT) the 1-inch-wide (25 mm) paper tape had 8 hole positions running across it, with the 8th being for parity. 7 positions were used for the data, e.g., 7-bit ASCII. The 8th position had a hole punched in it depending on the number of data holes punched.

For a contrary view, Seymour Cray, premier designer of supercomputers, held parity designs in contempt. He felt it showed poor design—if one designed a transmission path to be reliable, one would not have to waste resources on parity. His famous quote on this (circa 1963) was "Parity is for farmers" (after the use of the term "parity" in the New Deal). After he later included parity bits on the CDC 7600, Cray reputedly said that "I learned that a lot of farmers buy computers."[1]

See also [edit]

- BIP-8
- Parity function

References [edit]

1. ^ "Quotable Quotes by Seymour R. Cray" "The following UNVERIFIED quotes have been widely attributed to Seymour R. Cray." (dead)

External links [edit]

- Different methods of generating the parity bit, among other bit operations

Categories: Binary arithmetic | Data transmission | Error detection and correction | Parity (mathematics) | RAID