Bits of Assurance: Crafting Parity for Error Detection

Course Information:

Fall 2023

Main Instructor: Zahra Sadat Shariatmadar Mortazavi

Contact Via: zshariatmadar@aut.ac.ir

Main Teaching Assistant: Danial Fathpour

Contact Via: dfathpour@aut.ac.ir

Project Assistant: Seyyed Mohammad Hamidi

Contact Via: 1380hamidi@gmail.com

Introduction

Digital communication is an integral part of modern technology, enabling devices to exchange information swiftly and reliably. However, data corruption due to noise, interference, or other errors during transmission can lead to incorrect information being received. Error detection mechanisms are crucial for ensuring the integrity of data in communication systems.

This project introduces students to the concept of parity bits, a fundamental error detection technique. By implementing parity generation and checking mechanisms in Verilog, students will gain hands-on experience with digital design and error detection concepts, applicable to various fields such as computer hardware design, communications, and embedded systems.

Table Of Content

Project Usage and Benefits

The use of parity bits for error detection is widespread in computer memory, cache, data transmission, and storage systems. Learning to implement a parity bit generator and checker in Verilog provides students with the understanding of error detection principles and practices in digital systems design. This knowledge is vital for developing more reliable and robust digital systems that are essential in our increasingly data-driven world.

Real-World Application

In the real world, error detection is a foundational aspect of data transmission protocols. For example, network devices use parity bits to ensure that the data sent across cables is not corrupted. Memory modules use error-correcting codes (of which parity is a simple form) to detect and sometimes correct errors that occur due to electrical noise or other disturbances. Learning about parity bits equips students with the ability to apply such concepts to a multitude of applications, from designing fault-tolerant systems to optimizing communication protocols.

Project Description

Objective

The goal of this project is to design and implement two Verilog modules:

1. Parity Generator: A module that takes an 8-bit input vector and outputs a parity bit to ensure even parity.

2. Parity Checker: A module that accepts a 9-bit vector (8-bit input data plus parity bit) and determines whether the data adheres to even parity, indicating the presence or absence of an error.

Requirements

Parity Generator Module

- Input: 8-bit data vector.

- Output: 1-bit parity value.

- Functionality: The module must XOR all the bits of the input vector. If the number of '1's in the input vector is odd, the output should be '1' to achieve even parity. Otherwise, the output should be '0'.

Parity Checker Module

- Input: 8-bit data vector and a 1-bit parity.

- Output: 1-bit status indicating error detection.

- Functionality: The module must verify the 9-bit vector (8-bit data plus parity bit) for even parity. If the parity is even, the output should be '0' (no error). If the parity is odd, the output should be '1' (error detected).

Test Bench

Students must write a test bench for each module to validate its functionality. The test bench should provide various input vectors to the module and display the output to ensure correctness.

Examples of Input and Output

Parity Generator Example

- Input Vector: 8'b1011_0101

- Parity Bit (Output): 1'b1 (since there are five '1's, which is odd)

Parity Checker Example

- Input Vector: 8'b1011_0101

- Parity Bit: 1'b1

- Error Status (Output): 1'b0 (since the total number of '1's including the parity bit is six, which is even)

Bonus Challenges

Analysis of Parity Bit Method

Students are encouraged to analyze the parity bit method for error detection. This involves identifying the strengths and limitations of using parity bits. For example, while the parity bit method can detect any single-bit error, it fails to detect errors if two bits are flipped, resulting in the same parity as the original data.

Research on Alternative Solutions

As an additional challenge, students can explore alternative error detection and correction methods, such as checksums, cyclic redundancy checks (CRC), or more advanced error-correcting codes like Hamming codes. The students should compare these methods to parity bits in terms of reliability, complexity, and overhead.

Conclusion

This project is designed to provide students with a practical understanding of error detection in digital systems. By completing this project, students will not only learn about the implementation of parity generators and checkers but will also be encouraged to think critically about the limitations of such systems and explore more robust solutions. The skills acquired through this project are foundational for any future work in the field of digital design and communication systems.

Best Regards, Hamidi