7주차 예비보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20191583 이름: 김태곤

**1. Parity Bit 생성기 에 대해 조사하시오.**

패리티 비트 생성기는 데이터에 오류 감지용 비트, 즉 패리티 비트를 추가하는 장치나 소프트웨어를 말한다. 이 기술은 주로 데이터 전송이나 저장 과정에서 발생할 수 있는 오류를 감지하기 위해 사용된다. 패리티 비트를 통해, 데이터가 어디선가 전송되거나 저장될 때 데이터의 무결성을 검증할 수 있다.

패리티 비트 생성의 기본 원리는 데이터 비트열 중 1의 개수를 기반으로 한다. 여기에는 두 가지 주요 방식이 있다. 짝수 패리티(even parity)와 홀수 패리티(odd parity)이다. 짝수 패리티의 경우, 데이터 비트열 내의 1의 개수가 짝수가 되도록 패리티 비트가 설정된다. 예를 들어, 데이터 비트열 1010의 경우 1의 개수가 이미 짝수이므로 패리티 비트는 0이 된다. 하지만 1011의 경우 1의 개수를 짝수로 만들기 위해 패리티 비트를 1로 설정한다. 홀수 패리티의 경우에는 1의 개수가 홀수가 되도록 설정된다. 이는 짝수 패리티의 원리와 반대로 작동한다.

패리티 비트 생성기에서 XOR 게이트는 핵심적인 역할을 한다. XOR 게이트는 두 입력값이 다를 때만 1을 출력하는 논리 회로이다. 패리티 비트를 계산할 때, XOR 게이트를 사용하면 데이터 비트열 내의 1의 개수가 홀수인지 짝수인지를 판별할 수 있다. 만약 1의 개수가 짝수라면, XOR 게이트의 결과는 0이 되고, 홀수라면 1이 된다. 이는 짝수 패리티를 생성하는 데 직접적으로 사용될 수 있다.

홀수 패리티를 생성하기 위해서는 XOR 게이트의 출력에 NOT 게이트를 추가하면 된다. NOT 게이트는 입력된 비트를 반전시키는 역할을 하므로, XOR 게이트의 출력이 0이라면 1로, 1이라면 0으로 만들어 짝수 패리티 결과를 홀수 패리티로 변환할 수 있다.

아래 이미지는 각각 짝수 패리티 생성기와 홀수 패리티 생성기이다.

텍스트, 폰트, 도표, 스케치이(가) 표시된 사진

자동 생성된 설명 텍스트, 도표, 폰트, 스케치이(가) 표시된 사진

자동 생성된 설명

패리티 비트 생성기를 통해 생성된 패리티 비트는 데이터와 함께 전송되거나 저장된다. 수신자 측에서는 받은 데이터와 패리티 비트를 함께 검사하여 데이터의 무결성을 확인할 수 있다. 만약 데이터에 오류가 발생해 1의 개수가 예상과 달라진 경우, 패리티 비트를 통해 이를 감지할 수 있다. 그러나 이 방식은 단일 비트 오류는 감지할 수 있지만, 두 개 이상의 비트가 변경된 경우는 감지할 수 없다는 한계가 있다.

**2. Parity Bit 검사기 에 대해 조사하시오.[검사 부호 종류 포함]**

패리티 비트 검사기는 데이터 전송이나 저장 과정에서 발생할 수 있는 오류를 감지하기 위해 사용되는 장치 또는 알고리즘이다. 이 기술은 특히 오류가 자주 발생하는 통신 채널이나 저장 매체에서 중요하게 사용된다. 패리티 비트 검사기의 주요 목적은 데이터의 무결성을 확인하고, 가능한 오류를 식별하는 것이다.

패리티 비트 검사는 크게 짝수 패리티(even parity) 검사와 홀수 패리티(odd parity) 검사의 두 가지 유형으로 나뉜다.

짝수 패리티 검사 방식에서는 전송되거나 저장된 데이터 비트열 내의 1의 총 개수가 짝수가 되어야 한다. 수신자 측에서 데이터와 패리티 비트를 함께 검사하여 1의 개수가 짝수인지 확인한다. 짝수가 아니라면 오류가 발생한 것으로 간주한다.

홀수 패리티 검사는 1의 개수가 홀수가 되어야 한다는 점에서 짝수 패리티 검사와 반대이다. 수신된 데이터와 패리티 비트의 1의 총 개수가 홀수가 아닌 경우, 데이터 전송 중 오류가 발생했다고 판단한다.

아래 이미지는 4bit 짝수 패리티 검사기이다.

도표, 스케치, 라인 아트, 라인이(가) 표시된 사진

자동 생성된 설명

패리티 비트 검사기는 이러한 패리티 비트 설정 방식에 기반해 작동한다. 수신된 데이터와 패리티 비트를 검사하여 1의 총 개수가 설정된 패리티(짝수 또는 홀수)와 일치하는지 확인한다. 일치하지 않으면, 데이터 전송 중 오류가 발생했다고 판단할 수 있다. 이는 패리티 비트가 데이터의 무결성을 확인하는 간단하지만 효과적인 방법을 제공한다는 것을 의미한다.

**3. Parity Bit 검사기 외의 다른 오류 검출기 및 오류 정정기를 조사하시오.**

패리티 비트 검사 외에도 데이터 전송 오류를 감지하기 위한 여러 검사 부호 방식이 있다. 이러한 검사 부호들은 패리티 비트와 유사하게 데이터의 무결성을 보장하기 위해 설계되었으나, 각각 특성과 적용 방식이 다르다.

**1) CRC (Cyclic Redundancy Check)**

CRC (Cyclic Redundancy Check)는 데이터 전송 또는 저장 시 발생할 수 있는 오류를 검출하기 위해 널리 사용되는 방법 중 하나이다. 이 기술은 데이터의 무결성을 확인하고, 특히 네트워크 통신이나 저장 매체에서 데이터 손상을 감지하는 데 매우 효과적이다. CRC의 기본 원리는 데이터 블록을 특정 다항식으로 나누어 나머지를 계산하는 것에 기반한다. 이때 사용되는 다항식은 생성 다항식이라고 불리며, CRC 알고리즘의 핵심적인 부분을 구성한다.

CRC는 세 단계로 진행된다. 먼저 데이터 준비 단계로 전송하려는 데이터 블록 끝에 충분한 수의 0을 추가한다. 이는 생성 다항식의 차수에 따라 결정되며, 보통 생성 다항식의 차수보다 하나 적은 수의 0이 추가된다. 이후 나눗셈 단계로 수정된 데이터 블록을 생성 다항식으로 나눈다. 이 과정은 전통적인 수학적 나눗셈과는 다르며, 비트 연산(특히 XOR 연산)을 사용하여 수행된다. 최종적으로 나눗셈의 결과로 얻어진 나머지가 CRC 값(또는 CRC 체크섬)이 된다. 이 값은 원래의 데이터 블록에 추가되어 전송되거나 저장된다. 이러한 방식으로 생성된 CRC값은 다음 과정을 통해 오류를 검출한다. 수신자 측에서는 전송받은 데이터(원본 데이터 + CRC 값) 전체를 동일한 생성 다항식으로 나눈다. 이때 나머지가 0이면 데이터가 정확하게 전송되었다고 판단한다. 나머지가 0이 아니라면, 데이터 전송 과정에서 오류가 발생했다고 간주한다.

CRC는 데이터 블록 내에서 비트가 뒤바뀌는 경우나 비트가 누락되거나 추가되는 등의 오류를 잘 감지할 수 있다. 그러나, CRC는 특정 패턴의 오류를 감지하지 못하는 한계도 있으며, 모든 가능한 오류 상황을 감지할 수 있는 완벽한 방법은 아니다.

**2) 체크섬 (Checksum)**

체크섬은 데이터의 무결성을 검증하기 위해 사용되는 간단하면서도 효과적인 방법 중 하나이다. 이 기술은 데이터가 전송되거나 저장되는 동안 발생할 수 있는 오류를 감지하는 데 주로 사용된다. 체크섬의 기본 아이디어는 데이터를 구성하는 여러 부분의 합을 계산하고, 이 합을 사용하여 데이터의 정확성을 검사하는 것이다.

체크섬을 계산할 때, 데이터의 여러 부분(바이트나 워드)의 값을 모두 더하고, 이 합계가 특정 크기(8비트, 16비트)를 초과할 경우 오버플로우가 발생해 초과분을 무시하고 남은 값만을 사용한다. 때때로 합계를 반전시켜 최종 체크섬 값을 얻기도 한다. 계산된 체크섬은 원본 데이터와 함께 수신자에게 전송되며, 수신자는 받은 데이터에 대해 체크섬을 다시 계산하여 전송된 체크섬과 비교한다. 두 체크섬 값이 일치하면 데이터는 오류 없이 전송된 것으로 간주되고, 일치하지 않으면 데이터 전송 중 오류가 발생했다고 판단한다.

체크섬의 가장 큰 장점은 그 구현의 간단함과 계산의 효율성에 있다. 또한, 일반적인 오류 패턴을 효과적으로 감지할 수 있어, 많은 네트워크 프로토콜과 파일 시스템에서 기본적인 오류 검출 메커니즘으로 사용된다. 그러나 체크섬은 CRC나 오류 정정 코드(ECC)와 같은 보다 복잡한 기술에 비해 오류 감지 능력이 제한적이다.

**3) 해밍코드 (Hamming Code)**

해밍 코드의 핵심 아이디어는 데이터 비트 사이에 특정 규칙에 따라 추가적인 비트(해밍 비트)를 삽입하여, 데이터가 전송되거나 저장될 때 발생할 수 있는 단일 비트 오류를 검출하고 정정할 수 있도록 하는 것이다.

해밍 코드는 2의 거듭제곱 규칙을 따라 해밍 비트(오류 검출 및 정정 비트)를 배치한다. 즉, 1, 2, 4, 8, ... 위치에 해밍 비트가 삽입되며, 나머지 위치에는 실제 데이터 비트가 배치된다. 각 해밍 비트는 코드워드 내의 특정 비트들의 집합을 커버하며, 이들 비트의 합(1의 개수)이 짝수 또는 홀수가 되도록 설정된다(일반적으로 짝수 패리티가 사용된다).

예를 드어, 4비트 데이터에 대한 해밍 코드는 3개의 해밍 비트를 추가하여 총 7비트로 구성된다. 이 때, 각 해밍 비트는 데이터 비트와 다른 해밍 비트의 특정 조합을 통해 그 값이 결정된다.

이후 수신된 해밍 코드에서, 모든 해밍 비트에 대해 패리티 검사를 수행한다. 만약 모든 해밍 비트의 패리티 검사가 올바르다면, 데이터는 오류 없이 전송된 것으로 간주된다. 하지만 하나 이상의 해밍 비트에서 패리티 오류가 발견되면, 오류가 있는 비트의 위치는 오류가 발견된 해밍 비트들의 위치 번호를 합산하여 찾을 수 있다. 이를 통해 단일 비트 오류의 정확한 위치를 파악하고, 그 비트를 반전시킴으로써 오류를 정정할 수 있다.

해밍 코드의 주요 장점은 단일 비트 오류를 효율적으로 검출하고 정정할 수 있다는 것이다. 그러나 해밍 코드는 두 개 이상의 비트 오류에 대해서는 오류를 정확히 정정할 수 없다는 한계가 있다.

**4. N bit 비교기 에 대해서 조사하시오.**

N비트 비교기는 디지털 회로 설계에서 두 개의 N비트 값을 비교하여 그들의 관계(동일, A가 B보다 크거나 작음)를 결정하는 논리 회로이다. N비트 비교기는 기본적으로 두 개의 N비트 입력 A와 B를 받아 세 가지 출력 신호를 생성한다: A>B, A=B, A<B. 이 신호들은 입력된 두 값의 상대적인 크기를 나타낸다. N비트 비교기를 구현하는 방법에는 여러 가지가 있지만, 가장 일반적인 두 가지 접근 방식 연쇄 방식과 병렬 방식이다.

연쇄 방식(Chain Method)에서는 1비트 비교기를 기본 단위로 사용하여 N비트까지 확장한다. 각 1비트 비교기는 두 비트를 비교하고, 비교 결과를 다음 비트로 전달하여 최종적으로 전체 비트의 비교 결과를 도출한다. 연쇄 방식은 구현이 단순하지만, 비교를 위해 모든 비트를 순차적으로 처리해야 하므로 지연 시간이 길어질 수 있다.

병렬 방식(Parallel Method) 비교기는 N비트를 한 번에 비교할 수 있는 복잡한 논리 회로를 사용한다. 이 방식은 더 빠른 비교를 가능하게 하지만, 회로의 복잡성이 증가하고, 더 많은 하드웨어 자원을 요구한다.

N비트 비교기는 CPU의 명령어 집합 구현, 메모리 관리, 데이터베이스 시스템, 네트워크 장비 등 다양한 디지털 시스템 내에서 광범위하게 사용된다. 특히, 정렬, 최대/최소값 결정, 범위 검사 등의 연산에서 중요한 역할을 한다.

아래 이미지는 1 bit 비교기 예시이다.

**도표, 라인, 평면도, 종이접기이(가) 표시된 사진

자동 생성된 설명**

**5. IC 7485 비교기 에 대하여 조사하시오.**

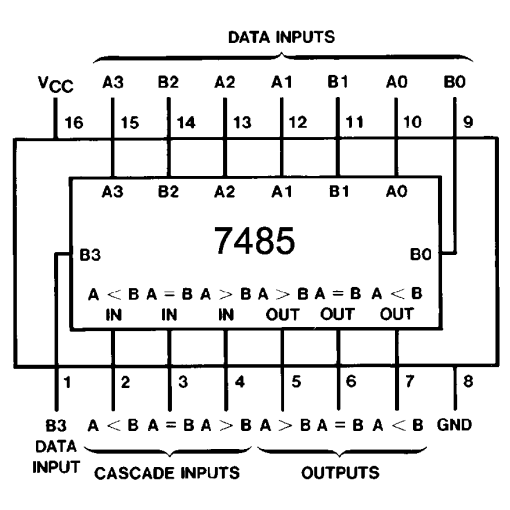
IC 7485는 4비트 magnitude 비교기로, 디지털 회로 설계에서 널리 사용되는 집적 회로(IC)이다. 이 IC는 두 개의 4비트 데이터(A0-A3, B0-B3)를 비교하여 A가 B보다 큰지, 작은지, 또는 두 값이 동일한지를 판단한다. IC 7485는 디지털 시스템에서 정렬, 데이터 비교, 순위 결정 등의 연산을 수행하는 데 중요한 역할을 한다.

C 7485는 한 번에 4비트씩 두 데이터 그룹을 비교할 수 있다. 이는 더 큰 비트 수의 데이터를 비교하기 위해 여러 IC를 직렬 또는 병렬로 연결하여 확장할 수 있음을 의미한다.

이 IC는 두 입력 그룹 A와 B를 비교하고, 그 결과를 세 가지 주요 출력 신호, 즉 AGBO (A Greater than B Output), AEBO (A Equal to B Output), ALBO (A Less than B Output)를 통해 표현한다. A>B, A = B, 그리고 A < B일 때 각각 1을 출력한다. 이 출력들은 비교되는 두 데이터 그룹의 상대적인 크기를 나타낸다.

여러 7485 IC를 연결하여 4비트를 초과하는 데이터 길이를 비교할 수도 있다. 이를 통해 8비트, 16비트, 또는 그 이상의 데이터 비교를 위한 더 큰 비교기를 구성할 수 있다.

아래 이미지는 7485 IC 4 bit magnitude 비교기 예시이다.



**6. 기타이론.**

3번 문제를 통해 데이터 전송이나 저장 시 오류를 검출하기 위해 사용되는 간단하면서도 효율적인 방법들을 알아보았다. 3번 예시 외에도 많은 오류 검출을 위한 방법들이 존재한다. 그중 LRC(Longitudinal Redundancy Check)에 대해 추가로 알아보자.

LRC는 주로 전송되는 데이터 블록의 각 열에 대한 패리티 비트를 계산하여 오류 검출 정보를 생성하는 데 사용된다. LRC는 특히 자기 테이프나 기타 순차적 데이터 저장 매체에서 데이터의 무결성을 검사하는 데 널리 사용되었다. LRC의 기본 원리는 전송하거나 저장할 데이터를 여러 개의 바이트(또는 단위 데이터 블록)로 구성된 행과 열의 형태로 구성한 후, 각 열에 대해 패리티 비트(짝수 패리티 또는 홀수 패리티)를 계산하는 것이다. 이렇게 계산된 패리티 비트들을 모아 새로운 바이트를 형성하고, 이를 LRC 바이트라고 한다. 이 LRC 바이트는 원래의 데이터 블록과 함께 전송되거나 저장된다. 이후 데이터 수신 시 또는 데이터를 읽을 때, 수신자는 동일한 방식으로 패리티 비트를 계산하여 LRC 바이트를 생성한다. 그런 다음, 전송된 LRC 바이트와 수신자가 계산한 LRC 바이트를 비교한다. 만약 두 LRC 바이트가 일치하지 않는다면, 데이터 전송 과정에서 오류가 발생했음을 의미한다. 이를 통해 데이터의 무결성을 검증할 수 있다.

LRC의 가장 큰 장점은 그 간단함과 구현의 용이성이다. 하드웨어나 소프트웨어 모두에서 쉽게 구현할 수 있으며, 추가적인 비용이 거의 들지 않는다. 그러나 LRC는 상대적으로 단순한 오류 검출 방식으로, 주로 단일 비트 오류나 짝수 수의 비트 오류를 검출하는 데 유효하다. 복잡한 오류 패턴이나 높은 신뢰성이 요구되는 환경에서는 CRC(Cyclic Redundancy Check)와 같은 더 복잡한 오류 검출 기술이 함께 사용될 수 있다.