7주차 예비보고서

전공: 아트&테크놀로지학과 학년: 4학년 학번: 20191048 이름: 김도솔

**1.**

**Parity Bit**는 데이터 전송 과정에서 발생하는 오류를 검출하기 위해 사용되는 비트이다. Parity Bit는 데이터 비트열에 추가되며, 이 비트의 값은 데이터 비트들 중에 있는 ‘1’ 비트의 수에 따라 결정된다. 따라서 데이터가 전송되는 동안 어떤 비트가 오류로 인해 변경되면, 패리티 비트의 값도 함께 변경되기에 비트 오류를 검출하는 것이 가능하다.

Parity Bit는 짝수 Parity Bit와 홀수 Parity Bit로 나뉘는데, 먼저 짝수 패리티 비트는 데이터 비트들 중에 ‘1’ 비트의 수가 짝수 개일 때 ‘1’로, 홀수 개이면 ‘0’으로 설정된다. 따라서 짝수 패리티 비트를 추가하면 전체 데이터 비트에서 ‘1’ 비트의 수가 항상 짝수가 된다. 예를 들어 7 비트 데이터 "0010110"을 전송하는 경우, 논리 1의 개수가 홀수이므로, 짝수 패리티를 유지하려면 패리티 비트를 1로 설정해야 한다. 따라서, 데이터 맨 앞에 짝수 패리티 비트를 조합한 결과는 "10010110"이 된다.

다음으로 홀수 패리티 비트는 데이터 비트들 중에 ‘1’ 비트의 수가 홀수 개일 때 ‘1’로, 짝수 개이면 ‘0’으로 설정된다. 따라서 홀수 패리티 비트를 추가하면 전체 데이터 비트에서 "1" 비트의 수가 항상 홀수가 된다. 예를 들어 7 비트 데이터 "0010110"을 전송하는 경우, 논리 1의 개수가 홀수이므로, 홀수 패리티를 유지하려면 패리티 비트를 0으로 설정해야 한다. 따라서, 데이터 맨 앞에 홀수 패리티 비트를 조합한 결과는 "00010110"이 된다.

이런 식으로 데이터에 Parity Bit를 추가하는 장치를 **Parity Bit 생성기**라고 한다. XOR 연산을 사용하면 데이터에서 ‘1’ 비트 수의 홀짝 여부를 알아낼 수 있기에, 이러한 XOR 연산의 특성을 활용해 Parity Bit 생성기를 구현할 수 있다. 짝수 패리티 비트 생성기는 입력된 n개의 이진 비트를 XOR 연산하여 패리티 비트를 생성한다. 따라서 (n-1)개의 XOR 게이트를 사용하여 구현할 수 있다. 홀수 패리티 비트 생성기는 입력된 n개의 이진 비트를 XOR 연산하고 그 출력에 보수를 취함으로써 패리티 비트를 얻는다. 이를 구현하기 위해서는 (n-1)개의 XOR 게이트와 하나의 NOT 게이트를 사용한다.

|  |  |  |
| --- | --- | --- |
|  | **3- bit even parity generator** | **3- bit odd parity generator** |
| Boolean Function |  |  |
| 회로도 | Even Parity Generator |  |

**2.**

**Parity bit 검사기**는 패리티 비트를 사용하여 데이터 오류를 감지하는 역할을 하는 장치이다. 패리티 비트 검사기는 다음과 같이 동작한다. 먼저 데이터를 받으면 수신된 데이터와 해당 데이터의 패리티 비트를 검사하여 오류 여부를 확인한다. 이때 패리티 비트 검사기는 데이터와 패리티 비트 간의 일관성을 확인하는데, 이는 송신 측에서 패리티 비트를 계산할 때 데이터 비트들의 짝수 또는 홀수 패리티를 유지하는 방식으로 계산하므로, 이 패리티가 수신 측에서도 일치해야 한다는 것을 의미한다. 만약 일치하지 않는 경우, 패리티 비트 검사기는 데이터에 오류가 있다고 판단하고 오류 신호를 발생시킨다. 오류를 검출한 경우 이를 상위 시스템에 보고하거나 적절한 조치를 취할 수 있는 인터페이스를 제공한다.

패리티 비트 검사기도 생성기와 마찬가지로 홀짝에 따라 나뉜다. 먼저 짝수 패리티 검사기는 전송된 데이터 중 짝수 패리티 비트를 포함한 전체 이진 비트의 논리 1의 개수가 항상 짝수여야 한다는 사실을 이용한다. 따라서, 수신 측에서 XOR 게이트를 사용해 데이터와 짝수 패리티 비트를 연산하여 출력이 0이 되면 데이터의 무결성이 유지된 것으로 간주한다. 출력이 1이면 전송된 데이터에 오류가 발생했다고 본다. 다음으로 홀수 패리티 검사기는 전송된 데이터 중 홀수 패리티 비트를 포함한 전체 이진 비트의 논리 1의 개수가 항상 홀수여야 한다는 사실을 이용한다. 따라서, 받는 측에서 XOR, NOT 게이트를 사용해 데이터와 홀수 패리티 비트를 연산하여 출력이 0이 되면 데이터의 무결성이 유지된 것으로 간주된다. 출력이 1이면 전송된 데이터에 오류가 발생했다고 본다.

패리티 비트 검사기는 간단하고 비용 효율적인 오류 검출 방법이지만, 단일 비트 에러만 감지할 수 있으며, 두 개 이상의 에러가 발생한 경우에는 에러를 식별할 수 없는 단점이 있다. 또한 오류를 감지할 뿐, 오류를 자동으로 수정하지는 않는다. 따라서 더 복잡한 오류 검출 및 수정 기술을 사용하려면, 패리티 비트 검사기를 보완하는 방식으로 해밍 코드(Hamming Code) 또는 CRC (Cyclic Redundancy Check)와 같은 고급 오류 검출 및 교정 기술을 사용할 수 있다.

|  |  |  |
| --- | --- | --- |
|  | **4- bit even parity checker** | **4- bit odd parity checker** |
| Boolean Function |  |  |
| 회로도 | Digital Combinational Circuits | Odd Parity Checker |

**3.**

**1) CRC (Cyclic Redundancy Check):** CRC는 데이터 전송 시 데이터의 무결성을 검사하기 위한 에러 검출 기술 중 하나이다. 데이터 송신 시 데이터 블록에 대한 다항식 값을 계산하고 이를 데이터 비트에 추가적인 비트(검사 비트 또는 CRC 체크섬)로 붙여 데이터를 전송하면 이 값을 수신 측에서 다시 계산하여 비교하는 방식으로 동작한다. 데이터 블록이 손상되었을 경우, CRC 값이 일치하지 않아 오류가 감지된다. CRC는 패리티 비트보다 더 복잡하지만 오류 검출 능력이 뛰어나고 하드웨어 상에서 구현하기 쉽다는 장점을 가진다.

**2) 해밍 코드 (Hamming Code):** 해밍 코드는 리차드 해밍(Richard Hamming)에 의해 개발되었으며, 데이터의 무결성을 확보하는 데 사용된다. parity bit 검사기와 다르게 해밍 코드를 이용하는 검사기는 데이터의 오류 검출 및 정정을 동시에 수행할 수 있다. 해밍 코드는 패리티 비트를 생성하기 위해 데이터 비트와 패리티 비트 간의 해밍 거리를 사용한다는 특징이 있으며 주로 컴퓨터 메모리, 디스크 저장, 데이터 전송 등의 응용에서 쓰인다.

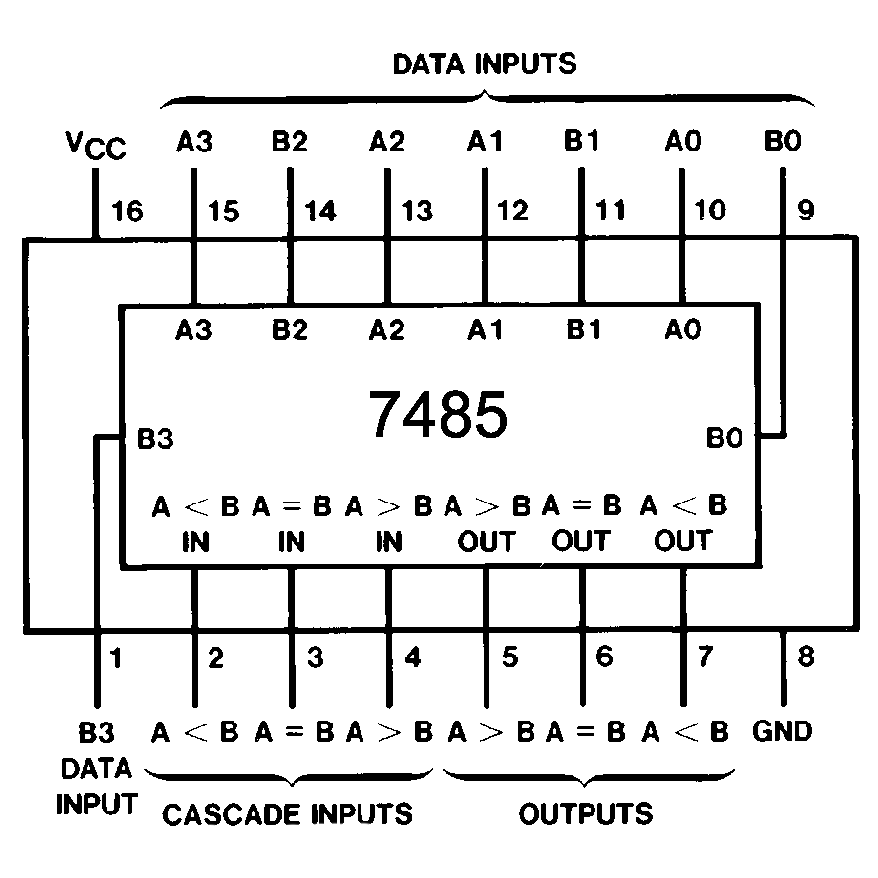
**3) FEC (Forward Error Correction):** FEC는 데이터 전송 시 데이터 오류를 실시간으로 검출하고 복구하는 기술이다. 이 기술은 주로 데이터가 손실되거나 왜곡될 수 있는 환경에서도 데이터의 손실을 방지하거나 복구하는 역할을 한다. FEC는 전송 데이터에 추가적인 리다운던트(불필요한) 데이터를 추가하고 이를 통해 오류를 검출하고 수정한다. 대표적인 FEC 기술로는 터보 코드(Turbo Codes)와 LDPC 코드(Low-Density Parity-Check Codes)가 있다.

**4.**

**N 비트 비교기**는 N 비트의 입력을 비교하여 두 개 이상의 값을 서로 비교하고 관련된 정보를 생성하는 장치이다. N 비트 비교기는 다음과 같이 동작한다. 먼저 N 비트 입력을 받으면 입력 값들을 서로 비교하여 논리적인 연산을 수행한다. 이때 주로 등호 비교(같음 여부), 크기 비교(크다/작다 여부), 논리 AND, OR, NOT 연산 등이 사용된다. 입력 값들 간의 비교 연산이 끝나면 그 결과에 따라 출력 정보를 생성한다. 일반적으로 두 입력이 같은 경우, 한 입력이 다른 입력보다 큰 경우, 논리 조건을 충족하는 경우 등 다양한 비교 결과를 나타내는 출력 비트를 생성한다. N 비트 비교기는 두 개 이상의 값을 비교할 수 있으며, 모든 입력 값에 대한 비교 결과를 출력으로 생성할 수 있다. 예를 들어, 4 비트 비교기는 4 개의 입력 값을 비교하고 4 개의 출력 비트를 생성할 수 있다. N 비트 비교기는 데이터베이스 정렬, 우선 순위 큐, 디지털 제어 시스템, 상태 검사 및 판별기, 주소 디코딩 등 다양한 응용 분야에서 사용된다.

**5.**

**IC 7485 비교기**는 4비트 이진 병렬 비교기(IC)로서, 7485 칩 또는 74LS85와 같은 이름으로도 알려져 있다. 이 IC는 TTL(Tansistor-Transistor Logic) 패밀리에 속하며, 데이터 비교 작업을 수행하기 위한 논리 회로를 내장하고 있다. IC 7485 비교기는 4비트 입력 A와 B를 받으면 입력 데이터 A와 B를 비교하여 여러 비교 연산을 수행하고, 연산이 완료되면 각 비트 별로 비교 결과를 출력하는 방식으로 동작한다. 좀 더 구체적으로 설명하자면, 라는 4비트 입력과 라는 4비트 입력을 비교한다. A가 B보다 클 때는 A>B를 1로, A가 B보다 작을 때는 A<B를 1로, A와 B가 서로 같을 때는 A=B를 1로 출력한다. 또한 IC 7485 비교기는 두 입력 A, B 뿐만 아니라 cascading input이라는 추가적인 입력을 받는다. 이때 cascading input은 하위 자릿수에서 올라오는 입력으로 4비트 비교기를 이용하여 더 큰 비트, 예를 들어 12비트, 16비트 비교기를 구현할 때 사용되는 입력이다. IC 7485는 비교 작업을 효과적으로 수행하기 위한 논리 회로를 내장하고 있으며, 복잡한 비교 작업을 수행하는 데 유용하다는 특징이 있다. 이는 주로 데이터 정렬, 우선 순위 결정, 상태 검사 및 판별, 디지털 제어 시스템 등 다양한 디지털 논리 응용 분야에서 사용된다.



[IC 7485 comparator]

**6.**

**1) 실수 비교기 (Floating-Point Comparator):** 실수 비교기는 부동 소수점 수를 비교하는 데 사용되는 비교기이다. 부동 소수점 형식은 실수를 표현하는 데 사용되며, 컴퓨터에서는 주로 IEEE 754 부동 소수점 표준을 따른다. 실수 비교기는 이러한 부동 소수점 형식에 따라 비트 패턴을 비교한다. 이때 비트 수준에서의 비교를 수행하며, 두 실수가 같은지 또는 크기 비교를 할 수 있다. 또한 오차 허용도를 설정하여 두 실수 간의 차이를 일정 범위 내에서 비교할 수 있다. 실수 비교기는 실수 계산에서 중요한 역할을 하며 제어 시스템이나 과학 및 공학 계산에서 사용된다.

**2) 비트 시프트 비교기 (Bit-Shift Comparator):** 비트 시프트 비교기는 입력된 두 비트 패턴을 비교하기 위해 사용되는 비교기이다. 주로 패턴 일치 또는 비트 패턴 내에서의 상대적인 위치를 판단하는 데 사용된다. 비트 시프트 비교기는 비트 시프트 연산을 사용하여 입력 비트 패턴을 이동한다. 이를 통해 입력된 비트 패턴의 일치 또는 비트 패턴 내에서의 상대적인 위치를 비교할 수 있다. 예를 들어, 데이터 검색 알고리즘에서 특정 비트 패턴을 찾는 데 사용될 수 있다.