7주차 예비보고서

전공: 아트엔테크놀로지 학년: 3학년 학번: 20191172 이름: 함승우

1.

Parity Bit 생성기는 오류를 검출할 때 사용하는 검사 부호 중 하나이다. 데이터를 전송할 때 오류가 생기면 데이터를 받는 쪽에서 어떤 오류가 생겼는지 모른다. 예를 들어, 데이터를 전송하는 쪽에서 7이라는 데이터를 보냈을 때, 오류가 생겨서 6이 되었다고 가정하자. 그래도 받는 쪽에서 이 6이라는 값이 정상적인 값인지 오류가 발생한 값인지 모른다. 따라서 이러한 정보 불균형을 해소하기 위해 parity 비트를 추가한다. 예를 들어, 홀수 비트의 전송을 원할 때 0110이라는 비트를 전송할 때, 뒤에 0의 비트를 붙이거나, 짝수 비트의 전송을 원할 때는 01101이라는 비트를 전송할 때 뒤에 1을 붙여서 전송한다. 이 데이터를 받는 수신자는 이러한 parity 비트를 확인해서 데이터의 오류를 확인할 수 있다. Parity 비트 생성기는 exclusive -or gate를 사용한다. Parity bit에는 두 가지 종류가 있는데, 짝수 parity와 홀수 parity가 있다. 이는 데이터를 보내기 전에 사전 약속을 해야 하고, odd parity의 경우에는 1의 개수를 홀수로 맞춰주어야 한다. Even parity는 1의 개수를 짝수개로 맞춰야 한다. 이는 exclusive – or과 non- exclusive or을 용도에 맞게 사용한다.

2.

마찬가지로 parity bit 검사기에는 짝수 검사가, 홀수 검사기가 있다. 짝수 검사기의 경우 결과 값이 0이면 데이터에 오류가 없다는 의미이고 1이 된다면 데이터 전송 도중 오류가 발생했음을 의미한다. 이 검사기의 검사 방법은 짝수 검사기는 전체 비트에서 1의 개수가 홀수이면 0, 1의 개수가 짝수이면 1을 출력한다. 반대로, 홀수 검사기의 경우에는 1의 전체 비트에서의 개수가 홀수면 1을 출력하고, 짝수면 0을 출력한다.

3.

첫번째로는 해밍 코드 방식이 있다. 이는 수신측에서 오류가 발생한 비트를 검출한 후 직접 수정하는 방식이다. 오류의 검출과 수정을 도맡아서 하게 된다. 1비트의 오류만이 수정이 가능하다.

텍스트, 스크린샷, 번호, 평행이(가) 표시된 사진

자동 생성된 설명

위는 해밍코드를 나타낸 표이다. 해밍코드는 예를 들어 4 비트의 데이터를 전송할 때 3가지의 체크 비트와 함께 보낸다. 각 체크비트와 데이터비트의 수식을 활용해서 에러 비트를 확인할 수 있다. 예를 들어, e1 = 1, e2 = 0, e4 = 1일 때 bit 5가 error임을 알 수 있다. 이 경우에는 4e4 + 2e2 + e1 비트를 조사해서 에러 비트인지 아닌지 확인하는 작업을 통해서 error를 검출하고 정정한다.

또한 순환 중복 검사가 있다. Cyclic Redundancy Check으로, 데이터에 오류가 발생했는지 확인하는 코드를 데이터 뒤에 확장 데이터를 덧붙여 보내는 방식이다. 원본 데이터에 다항식 비트수 - 1만큼의 0을 추가하고, XOR 연산을 계속하는 방식으로 해당 비트값의 검증값을 확인한다.

4.

우선 비교기란, 여러 개의 값을 서로 비교하는 연산자이다. 1-bit 비교기의 경우에는 이전 실습에서 진행한 적이 있는데, A>B일 경우에 Output을 AB’를 출력하고, A=B인 경우에는 A XNOR B를 출력, A<B인 경우에는 A’B로 구현할 수 있다.

N bit 비교기는 N개의 비트 두 개를 입력 받는다. 이 때 input의 크기는 총 2N개가 된다. 이때 비교기는 1 bit 비교기를 서로 조합하여 큰 비교기를 설계할 수 있다. 예를 들어, 3비트 비교기는 1 비트 비교기와 2비트 비교기를 서로 조합하여 얻을 수 있다.

| **InPut** | | | | **OutPut** | | |
| --- | --- | --- | --- | --- | --- | --- |
| **A1** | **A0** | **B1** | **B0** | **A<B** | **A=B** | **A>B** |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

위는 2 bit 비교기를 나타낸 모습으로, 이 진리표를 사용하여 K-map을 구성하고 Boolean 식을 구성할 수 있다.

A > B의 경우에는 A1B1’ + A0B1’B0 + A1A0B0’ 이고 A = B의 경우에는 (A0 XNOR B0)(A1 XNOR B1)이다. A<B의 경우에는 A1’B1 + A0’B1B0 + A1’A0’B0이다.

5.

IC 7485 비교기는 4비트 크기의 이진수 두 개를 서로 비교한 결과를 출력하는 회로이다.

이 직접 회로 칩은 보통 4비트 입력 두 개 값을 받는 것과는 다르게 3개의 입력을 추가로 받게 된다. 각각 이전 단계에서의 출력을 입력으로 받게 된다. 마찬가지로 3개의 output을 갖고 그는 각각 A>B, A=B, A<B의 경우에 해당한다. 앞에서 말했듯이 이전 단계에서의 A>B, A=B, A<B도 입력값으로 받는데 현재 비교하는 단계에서의 결과가 같다면 하위 비트의 비교 결과로 수를 비교하게 된다. 4비트 수 2, A와 B를 비교하고자 한다면, A>B, A=B, A<B(이전 단계에서의 결과), A0, A1, A2, A3, B1, B2, B3, B4 가 입력이 되고, 결과값은 A>B, A=B, A<B가 출력된다.

6.

중복 검사의 한 형태인 체크 섬은 나열된 데이터를 모두 더하여 체크섬 숫자를 얻는다. 또한 이러한 checksum을 정해진 비트 수의 모듈러로 정해진 비트 수로 재구성하게 된다. 만약, 체크섬이 맞아 떨어지지 않는다면 데이터가 손상되었다고 판단한다. 예를 들어 4바이트의 데이터 0x25, 0x62, 0x3F, 0x52가 있다고 가정하자. 이 때 이 데이터들을 모두 더하면 0x118이 된다. 이 때 자리올림 수 인 1을 제외하면(캐리니블)을 삭제하면, 0x18이 된다. 이때 캐리니블은 최상위 니블이 되기 때문에 0x118을 2진수로 변환시킨 0001 0001 1000 중 맨 앞의 0001을 삭제하면 된다. 그 후 2의 보수법을 진행하면 0001 1000에서 1110 1000이 된다. 이를 16진수로 다시 나타내면 0xE8이 된다. 이를 체크섬이라 칭한다. 순서를 정리하자면 1)모두 더하고, 2)캐리니블 삭제, 3)2의 보수법을 적용시키면 체크섬을 구할 수 있다. 이 체크섬을 다시 모두 더한 값인 0x118에 더해야 한다. 더 하게 되면 0x200이고 캐리니블을 또 삭제하면 0000 0000이 되는데, 이는 오류가 없음을 의미한다.