7주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20211522 이름: 김정환

**1.**

.................

Parity Bit는 데이터를 송수신하면서 binary information에 오류가 발생했는지 확인하기 위해 사용한다. 이러한 과정없이 데이터를 전송하면, 데이터를 받는 사람은 중간에 오류가 발생했는지 알 수 없기 때문이다. 따라서 Parity Bit는 1 bit를 추가하여 보내는 방식을 통해서 오류를 확인한다. 여기에는 2가지 종류가 있는데, 짝수 parity와 홀수 parity가 존재한다. 이는 보내는 데이터에서 Parity Bit를 포함하였을 때 1의 개수가 짝수인지 홀수인지에 따라 구분한다. 또한 Parity Bit가 추가되므로 3-bit Parity Generator는 4-bit Parity Checker에 의해 확인된다. 따라서 Parity Generator는 보낼 데이터들을 입력으로 받아 추가될 Parity Bit의 값을 출력하는 회로이다. 짝수 parity인 경우에는 1이 홀수 개이면 1을 추가하므로 XOR Gate를 사용하여 구현하고, 홀수 parity의 경우에는 1이 짝수 개이면 1을 추가하므로 XNOR Gate를 사용하여 구현한다.

................

**2.**

.......................

Parity Bit Checker는 Parity Bit가 포함되어 들어온 데이터의 1의 개수를 통해서 오류 여부를 확인하는 회로이다. 이는 Parity Generator와 유사하게 1의 개수가 짝수인지 홀수인지를 검사하는 방식으로 구현된다. Parity Bit를 포함한 개수만큼의 bit로 판별하므로 1 bit 적은 Generator가 만든 데이터를 검사하게 된다. 짝수 parity의 경우에는 1이 짝수 개 있을 때 참이므로 XNOR Gate를 사용하여 구현한다. 홀수 parity의 경우에는 1이 홀수 개 있을 때 참을 출력하므로 XOR Gate를 사용하여 확인한다.

.........................

**3.**

.......................

데이터의 오류를 검출하는 방식 중 checksum에 대해 조사하였다. 이는 모든 데이터의 값을 더한 후에 1의 보수를 취한 값을 사용하는데, 이 값을 데이터를 모두 더한 값과 비교하게 된다. 데이터에 오류가 없이 전달되었다면, 이는 모두 1이 나오게 되는데, 여기서 0이 나오게 되었을 때 오류가 있다고 판단하게 된다. 이러한 원리와 비슷하게 사용하여 Checksum 방식에는 여러 가지가 있지만 이 중에서 위처럼 단순하게 모두 더하는 방식은 단순 합계 체크섬이고, 이 방식을 사용했을 때 발생할 수 있는 오버플로우를 회피하기 위해 추가적인 조치를 취한 방식이 있다.

오버플로우를 회피하기 위한 방법으로 캐리, 즉 제일 앞의 4 bit를 버리는 방식이 있다. 이 방식으로 예시를 들어보면 데이터의 합이 500일 때 이는 binary로 0001 1111 0100이 되는데 제일 앞의 0001을 버리는 것을 의미한다. 이 후에 생성하는 것에는 위처럼 1의 보수 후에 OR 연산으로 확인하는 방법도 있지만, 2의 보수를 취하고, 이를 데이터의 합과 더하여 모두 0이 나오는지 확인하는 방법도 있다.

.........................

**4.**

.......................

비교기는 두 수를 비교하여 큰 쪽을 알려주거나 같은지 여부를 알려주는 회로이다. N bit 비교기 전에 우선 1 bit 비교기 먼저 보면, 두 수 A, B를 입력으로 받고, 출력은 EQ, LT, GT로 3개의 출력을 가진다. LT, GT는 A를 기준으로 하였고, 이를 각각 Verilog의 식으로 나타내면, EQ=~(A^B), LT=(~A)B, GT=A(~B)로 나타낼 수 있다. 다음으로 2 bit에 대해서는 카르노 맵을 활용하여 논리식을 정리하여 구현할 수 있고, 4비트의 경우 두 수의 대소관계 비교를 위해서는 모든 비트가 같아야 같은 수이고, 한 쪽이 큰 bit가 먼저 나오는 쪽이 큰 수임인 관계를 바탕으로 구현한다. 이와 1 bit 비교 때의 식을 응용하여 구성하면, 아래와 같이 4 bit 비교기를 구현할 수 있다.

도표, 평면도, 라인, 개략도이(가) 표시된 사진

자동 생성된 설명

가산기를 구현할 때 Full Adder 여러 개를 연결하여 여러 bit의 연산으로 확장하던 것과 마찬가지로 비교기도 확장할 수 있다. 따라서 4 bit 비교기를 엮어서 원하는 N-bit 비교기를 구현한다.

.........................

**5.**

.......................

IC 7485 비교기는 4 bit 비교기로 4 bit 수 2개의 크기를 비교하여 결과를 알려주는 회로이다. 여기에는 4 bit 두 개를 위한 input 단자 외에도 그 이상의 데이터 크기 비교를 위한 input 단자도 존재한다. 이 input 단자는 AGBI, ALBI, AEBI의 3개로 이는 이전 단계에서의 출력을 이어받는 부분이다. 출력 단자는 AGBO, ALBO, AEBO로 순서대로 A greater than B, A less than B, B equal B로 이 표현은 위의 input 단자에서도 똑같이 적용된다. 이 단자들을 통해서 4 bit 두 개가 서로 같을 때도 해당 input 3개를 통해서 크기를 판정할 수 있다. 또한 이들을 연결하여 더 큰 bit의 비교기를 구현할 수 있다.

텍스트, 폰트, 라인, 도표이(가) 표시된 사진

자동 생성된 설명

위의 그림이 IC 7485를 4개 연결하여 16 bit 비교기를 구현한 것이다.

.........................

**6.**

.......................

Parity Bit는 오류가 발생했다는 것만 확인할 수 있고 2개의 bit에서 오류가 발생시에는 오류를 검출하지 못한다. 이를 보완하여 해밍 코드를 사용하는데, 이는 Parity Bit를 데이터의 비트 수에 따라 필요한 만큼 추가하고 이를 조합하여 에러 검출과 교정을 수행한다. 필요한 Parity Bit의 개수는 아래의 식을 통해 얻을 수 있다.

2^p >= d + p + 1 (p : Parity Bit 수, d : 데이터 bit 수)

또한 Parity Bit는 2의 거듭제곱에 해당하는 위치에 삽입된다. 예제로 4 bit에 대해서 짝수 parity로 해밍 코드를 생성하는 과정은 다음과 같다. 4 bit인 경우에는 위의 식을 이용하면 Parity Bit가 적어도 3개는 필요하므로 각각을 p1, p2, p3로 나타내고, 데이터는 d1, d2, d3, d4로 나타내본다. 그러면 Parity Bit가 삽입되었을 경우, p1,p2,d1,p3,d2,d3,d4의 순서대로 Bit가 배치된다. 그 후 각각의 p1, p2, p3와 데이터의 값을 연산함으로써 오류를 검출하고 3개의 연산 결과에서 오류를 찾은 것을 바탕으로 오류의 위치를 찾는다.

.........................