7주차 예비보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20191629 이름: 이주헌

1. Parity bit 생성기에 대해 조사하시오.

Parity bit는 어떠한 이진수 코드에 추가할 수 있는 오류 검사 비트를 말한다. 송수신된 데이터에 오류가 있는지 여부를 검사하기 때문에 check bit라는 이름으로도 불린다. Parity bit는 크게 두 가지 종류로 나눌 수 있는데, 먼저 어떤 데이터 조각 안의 1의 개수를 짝수로 만드는 even parity와 홀수로 만드는 odd parity로 나눌 수 있다.

예를 들어, 어떤 신호에 1바이트의 데이터를 담아 보낼 수 있다고 하자. 이 신호에 0100010의 7비트 데이터를 담아 보내려고 할 때, even parity 체계를 사용하려면 이미 데이터를 담는 7비트에 1이 두 개 들어가 있으므로 데이터 비트 스트림의 끝에 0을 추가하여 송신하게 된다. 반대로, odd parity 체계를 사용하면 1의 개수를 홀수로 만들기 위해 데이터 비트 스트림의 끝에 1을 추가하여 송신하게 된다.

Parity 비트를 생성하는 방법은 1이 짝수 개 있는지 홀수 개 있는지 판별하는 회로를 사용하면 쉽게 구현할 수 있다. Even parity의 경우 1이 홀수 개 존재하면 1을 추가하고, 그렇지 않은 경우 0을 추가하기 때문에 XOR 게이트를 사용해서 구현할 수 있고, 반대로 odd parity의 경우 1이 짝수 개 존재하면 1을 추가하고, 그렇지 않은 경우 0을 추가하기 때문에 XNOR 게이트를 사용해서 구현할 수 있다.

2. Parity bit 검사기에 대해 조사하시오.

Parity bit에 두 종류가 존재하기 때문에 parity bit 검사기 역시 두 종류가 존재한다. Parity bit를 생성하는 방법과 마찬가지로, parity bit를 검사할 때는 읽어들인 데이터 비트에 1이 짝수 개 있는지 홀수 개 있는지 검사하는 회로가 사용된다. 읽어들인 데이터가 even parity를 사용하여 검증된 데이터라면, 모든 입력을 XNOR 게이트에 넣어서 1이 짝수 개 있는지 확인할 수 있고, 반대로 odd parity를 사용하여 검증된 데이터라면, 모든 입력을 XOR 게이트에 넣어서 1이 홀수 개 있는지 확인할 수 있다.

3. Parity bit 검사기 외의 다른 오류 검출기 및 오류 정정기를 조사하시오.

보통 인터넷에서 소프트웨어를 다운로드 받을 때 프로그램의 오류를 검출하는 방식은 메시지 다이제스트(message digest)를 비교하는 방식이다. 이 방식은 암호화 해시 함수를 사용하여 받은 데이터가 완전히 같은지 검증하는 방식이다. 여기서 암호화 해시 함수는 가변 길이의 데이터를 입력받아 고정된 길이의 비트 스트림을 생성하는 알고리즘을 통틀어 이르는 말으로, 가장 널리 사용되는 해시 함수로는 MD5[1], SHA-1[2], SHA-256 등이 있다. 데이터를 보내는 측에서 미리 데이터의 다이제스트를 공개하면, 데이터를 받는 측에서 데이터를 전부 수신한 뒤 다이제스트를 계산하여 송신자의 다이제스트와 수동으로 비교하는 방식을 사용하는데, 암호화 해시 알고리즘은 쇄도 효과(avalanche effect)에 의하여 단 1비트 차이도 해시값에 큰 영향을 주므로 데이터의 신빙성을 보장할 수 있다.

이러한 오류 검출기를 사용하면 송신자와 수신자 사이의 통신의 신빙성을 보장하는 오류 정정 통신 방식을 구성할 수 있는데, 이를 automatic repeat request(ARQ)라고 부른다. ARQ에서는 데이터를 프레임 단위로 잘게 잘라서 구성하게 되는데, 각 프레임을 생성하여 송신자가 수신자에게 전송하면, 수신자는 받은 데이터 프레임을 parity bit나 checksum과 같은 오류 검출기를 사용하여 검증하고, 검증 결과를 다시 송신자에게 보낸다. 만약 올바른 정보가 수신되었다면 (ACKnowledge) 다른 데이터 프레임을 수신하도록 송신자에게 전달하고, 올바르지 않은 정보가 수신되었다면 (Negative ACKnowledge) 방금 전송한 데이터를 다시 전송하도록 송신자에게 전달하는 것이다. 이 방법을 사용하는 가장 대표적인 통신 프로토콜은 인터넷 연결을 책임지는 TCP 프로토콜이다.

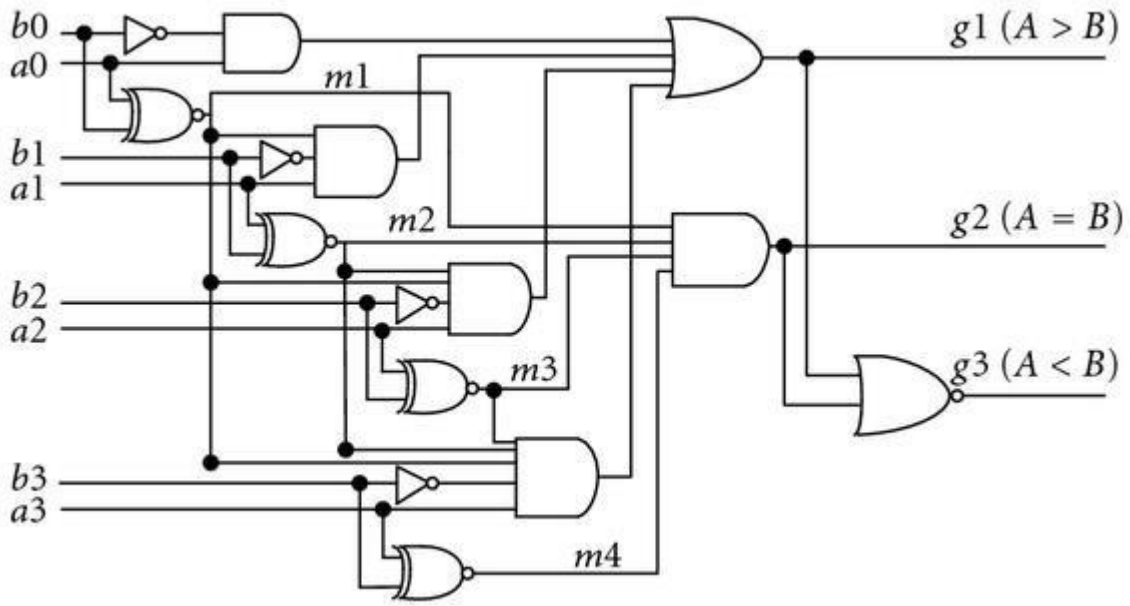
4. N-bit 비교기에 대해서 조사하시오.

1비트 비교기는 이미 5주차 실습에서 구현해본 바 있다. 또한, 1비트 비교기를 구현했을 때와 마찬가지로 각 입력에 대한 진리표를 작성하고, 진리표로부터 논리식을 도출하여 1비트 이상의 입력을 받을 수 있는 비교기를 설계하는 것 또한 가능하다. 예를 들어, 4비트 비교기는 다음과 같은 논리회로를 이용하여 구현할 수 있다.

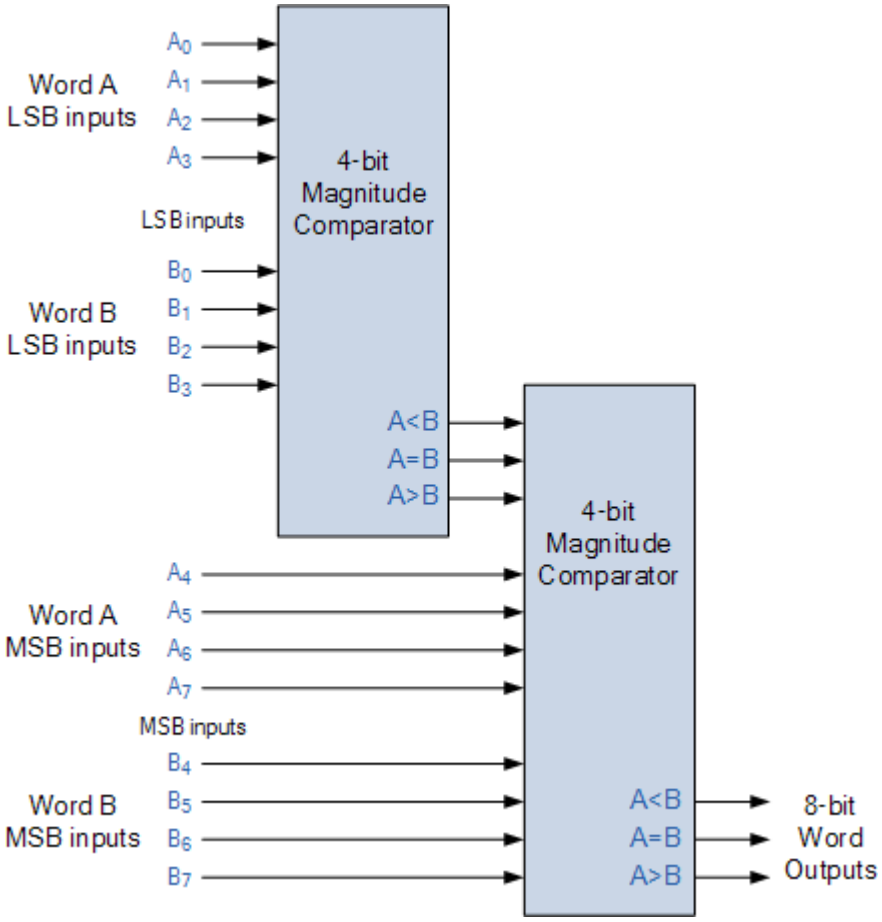
---

[1] 1996년 첫 해시 충돌이 보고된 이래로 2013년 완전히 같은 해시를 갖는 비트 스트림을 찾는 방법이 발견되면서 자주 사용되지 않는다.

[2] Git 버전 관리 소프트웨어에서 하나의 커밋을 구성할 때 사용되는 알고리즘으로, 이 알고리즘 역시 2017년 해시 충돌이 발견된 이후 자주 사용되지 않는다.



그러나 부울 방정식을 푸는 것은 쉬운 과정이 아니기 때문에 이런 방식으로 만들 수 있는 논리 회로는 한계가 있다. 따라서, 전가산기를 엮어서 N-bit 가산기를 만들었던 것과 같이, 이번에도 여러 4비트 비교기를 엮어서 더 큰 입력을 받을 수 있는 비교기를 설계할 수 있다. 이와 같은 비교기 디자인을 cascade comparator라고 부른다. 예를 들어, 1바이트 비교기는 다음과 같이 4바이트 비교기 2개를 연결하여 구성할 수 있다.

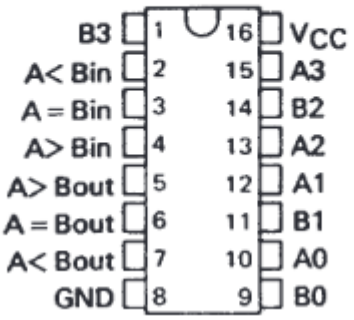


5. IC 7485 비교기에 대하여 조사하시오.

7485 집적 회로 칩은 바로 위 그림에서 *4-bit Magnitude Comparator*라고 쓰여진 회로를 구현하는 집적 회로 칩이다. 이 4비트 비교기는 4비트 입력 두 개 이외에도 3개의 입력을 더 받는데, 각각 이전 단계에서의 출력을 입력으로 받을 수 있도록 되어 있다. 최하위 비트를 시작으로 상위 비트를 계산하면서, 하위 비트의 연산 결과를 상위 비트로 "흘려보낼 수 있는" 회로가 7485 IC이다. 아래는 전자 부품 제조 회사인 Texas Instruments에서 제조하는 7485 비교기 구현체인 SN7485의 핀 다이어그램이다.[3]

[3] 보통 서로 다른 제조사라도 같은 논리회로를 구현하는 소자라면 핀 위치를 통일하는 경우가 많기 때문에 TI 뿐만 아니라 다른 제조사도 같은 핀 다이어그램을 공유한다고 볼 수 있다.

SN5485, SN54LS85, SN54S85 . . . J OR W PACKAGE
SN7485 . . . N PACKAGE
SN74LS85, SN74S85 . . . D OR N PACKAGE
(TOP VIEW)



6. 기타이론

이미 앞에서 조사한 parity bit를 이용해서도 오류를 검출하고, 거기다 정정까지 할 수 있는 데이터 표현 방식을 만드는 것이 가능하다. 이러한 parity bit의 이용법을 해밍 부호(Hamming's code)라고 부른다. 해밍 부호는 기본적으로 여러 개의 parity bit를 전략적인 위치에 삽입하여 오류가 발생한 위치의 비트를 특정할 수 있는 방식을 사용하고 있다. 이러한 "전략적 위치"를 알아내는 알고리즘은 다음과 같다.

1. 데이터의 비트에 전부 1부터 순서를 매긴다. (1번째 비트, 2번째 비트, 3번째 비트, ...)
2. 비트 순서를 이진수로 나타낸다. (1번째 비트, 10번째 비트, 11번째 비트, ...)
3. 비트 순서 안 1의 개수가 하나인 비트(즉, 비트 숫자가 2의 제곱수인 비트)는 모두 parity bit로 설정한다.
4. 3번 단계에서 parity bit로 설정되지 않은 모든 비트는 데이터 비트이다.
5. 모든 데이터 비트는 다음과 방식을 따라 결정된 두 개 이상의 parity bit의 집합에 의해 검증된다. 각 parity bit의 집합은 유일하다.
    a. 1번 parity bit는 비트 순서의 최하위 비트가 1으로 설정된 모든 비트를 검증한다. (1번째 비트, 11번째 비트, 101번째 비트, 111번째 비트, ...)
    b. 2번 parity bit는 비트 순서의 두 번째 비트가 1으로 설정된 모든 비트를 검증한다. (10번째 비트, 11번째 비트, 110번째 비트, 111번째 비트, ...)
    c. 4번 parity bit는 비트 순서의 세 번째 비트가 1으로 설정된 모든 비트를 검증한다. (100번째 비트, 101번째 비트, 1100번째 비트, 10101번째 비트, ...)
    d. 이를 일반화하면, 각 parity bit는 parity bit의 순서와 비트 순서의 논리곱을 계산했을 때 0이 아닌 값이 나오는 모든 비트를 검증한다.

예를 들어, 10011010이라는 1바이트 데이터에 해밍 부호를 삽입하고자 한다고 하자. 그러면 parity bit를 삽입한 뒤 비트 스트림은 PP1P001P1010이 될 것이다. 이제 각 parity bit에 올바른 정보를 채워넣으면 011100101010이라는 비트 스트림이 나오게 된다.