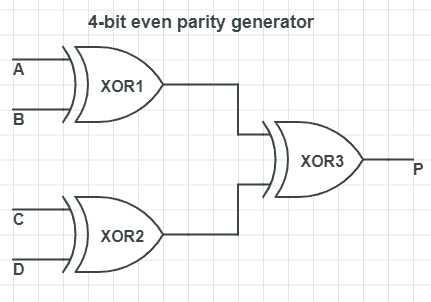
7주차 예비보고서

전공: 물리학과 학년: 3학년 학번: 20191286 이름: 김나현

1. Parity Bit 생성기에 대해 조사하시오.

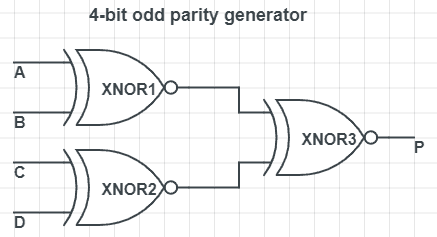
Parity bit란 이진 비트를 전송할 때, 한 비트가 달라지는 single error, 즉 단일 에러 검출을 위해 사용되는 비트를 의미합니다. Parity bit는 데이터에 포함된 논리 1의 개수를 세어 그 수가 짝수인지, 홀수인지에 따라 결정되는데 even parity bit를 사용하면 논리 1의 개수가 짝수일 때 parity bit를 0으로 설정하고 홀수일 때 parity bit를 1로 설정하고 반대로 odd parity bit를 사용하면 논리 1의 개수가 홀수일 때 parity bit를 0으로, 짝수일 때 1로 설정합니다. 이렇게 하면 even parity bit를 사용할 때는 parity bit를 포함한 전체 이진 비트 중 논리 1의 개수가 항상 짝수가 되고, odd parity bit를 사용할 때는 parity bit를 포함한 전체 이진 비트 중 논리 1의 개수가 항상 홀수가 됩니다.

따라서, parity bit를 포함하지 않은, 원래 보내려는 이진 비트 중 논리 1의 개수가 홀수일 때는 even parity bit가 1이고, 짝수일 때는 even parity bit가 0이므로 n개의 이진 비트가 있다고 할 때 even parity bit을 결정하기 위해서는 XOR 게이트를 사용하여 EVEN\_P(even parity bit)=An⊕An-1⊕…⊕A1으로 나타낼 수 있습니다. 따라서, even parity bit 생성기는 입력 n개를 XOR 연산하여 parity bit를 생성하므로 (n-1)개의 XOR 게이트를 이용하여 구현됩니다.



<Figure 1> 4 bit even parity bit 생성기

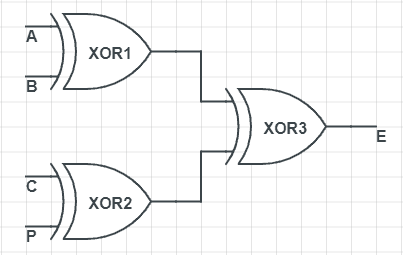
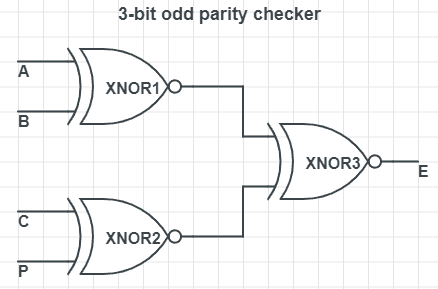
반대로, parity bit를 포함하지 않은, 원래 보내려는 이진 비트 중 논리 1의 개수가 짝수일 때는 odd parity bit가 1이고, 홀수일 때는 odd parity bit가 0이므로 n개의 이진 비트가 있다고 할 때 odd parity bit을 결정하기 위해서는 XNOR 게이트를 사용하여 ODD\_P(odd parity bit)=(An⊕An-1⊕…⊕A1)’으로 나타낼 수 있습니다. 따라서, odd parity bit는 입력 n개를 XOR 연산한 후, 그 출력에 보수를 취해서 얻을 수 있으므로 odd parity bit 생성기는 (n-1)개의 XOR 게이트와 하나의 NOT 게이트를 이용해 구현되거나 (n-2)개의 XOR 게이트와 하나의 XNOR 게이트를 이용해 구현되거나 (n-1)개의 XNOR 게이트로 구현됩니다.



<Figure 2> 4 bit odd parity bit 생성기

1. Parity Bit 검사기에 대해 조사하시오(검사 부호 종류 포함).

입력의 개수가 n개일 때, 모든 입력 중 참인 입력의 개수가 홀수일 때 1을 출력하는 XOR 게이트의 특성과 참인 입력의 개수가 짝수일 때 1을 출력하는 XNOR 게이트의 특성을 이용하여 parity bit 검사기를 구현할 수 있습니다.

전송된 데이터 중 even parity bit를 포함한 모든 이진 비트 중 논리 1의 개수가 항상 짝수여야 하므로 반대로 데이터를 받는 receiver는 XOR 게이트를 이용하여 even parity bit를 포함한 모든 비트를 연산하였을 때 출력이 0이 되면 데이터의 훼손이나 에러 없이 제대로 전송이 되었다는 것을 의미하고, 출력이 1이 되면 전송 받은 데이터에 에러가 존재한다는 것을 의미합니다. 반대로 odd parity bit를 포함한 모든 이진 비트는 논리 1의 개수가 항상 홀수여야 하므로 receiver는 XNOR 게이트를 이용하여 odd parity bit를 포함한 모든 비트를 연산하였을 때 출력이 0이 되면 전송 받은 데이터에 에러가 없다는 것을 의미하고, 출력이 1이 되면 전송 받은 데이터에 에러가 존재한다는 것을 의미합니다.

<Figure 2> 3 bit even parity bit 검사기와 3 bit odd parity bit 검사기

하지만 parity bit 검사기는 짝수 개의 에러가 발생한 경우, 에러가 발생했다는 것을 알 수 없고 에러가 발생해도 어느 부분에서 에러가 발생한 것인지 알 수 없다는 단점이 있습니다.

1. Parity Bit 검사기 외의 다른 오류 검출기 및 오류 정정기를 조사하시오.

해밍 코드(Hamming code)란 이진 선형 블록 오류 정정 부호의 일종으로, 어느 부분에서 에러가 발생했는지 알 수 없는 parity bit 검사기와 달리 해밍 코드를 이용한 검사기는 1비트의 에러를 정정할 수 있다는 점에서 차이가 있습니다. 송신한 데이터와 수신한 데이터 간의 차이가 다른 비트의 수를 해밍 거리(Hamming distance)라는 용어로 정의하며 전송하는 2진수 중 1,2,4,..,2^n번째 비트를 parity bit로 사용합니다. 따라서, 2^p bit 데이터가 수신되면 (p+1)개의 bit는 parity bit이고 나머지 (2^p-p-1)개의 bit는 원래 송신하고자 한 데이터일 것입니다. 구체적으로 알아보기 위해 4비트의 이진수를 전송할 때, 3개의 parity bit를 사용하여 총 7비트를 부호화하여 전송하는 상황을 예로 들겠습니다. 10진수 5를 전송하고자 할 때, 즉 0101의 4 bit 데이터를 전송하려고 할 때 3개의 parity bit를 사용하게 되고 송신하고자 하는 데이터가 C1, C2, C3, C4, C5, C6, C7이라고 할 때, 각 비트에 대해 다음과 같이 나타낼 수 있습니다.

C1 C2 C3 C4 C5 C6 C7

0 1 0 1

7개의 데이터가 송신할 것이므로 첫 번째, 두 번째, 네 번째는 parity bit이고 C1은 3, 5, 7번째의 있는 값들의 짝수 parity 값이고, C2는 3, 6, 7번째에 있는 값들의 짝수 parity 값, C4는 5, 6, 7번째의 짝수 parity 값입니다. 따라서, C1은 0이고, C2는 1, C4는 0이 됩니다. 따라서, 위의 경우에 대해 정확한 해밍 코드는 0100101입니다. 하지만 송신하는 과정에서 noise로 인해 error가 발생하여 C5가 0이 되면 수신기에서 수신한 해밍 코드는 0100001일 것입니다.

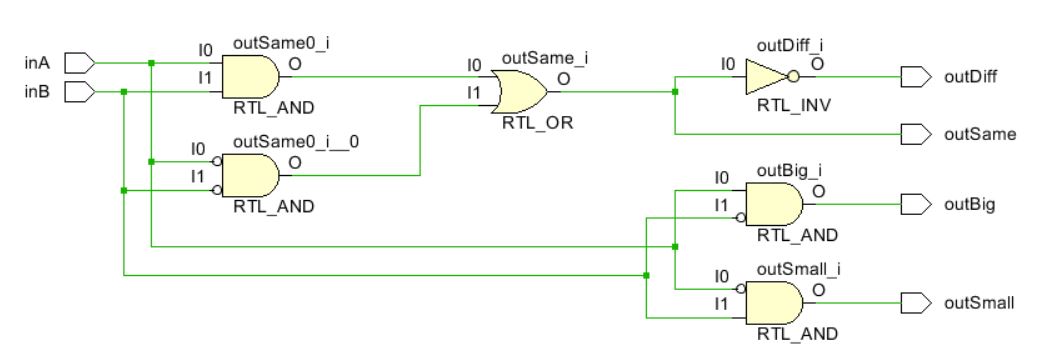
C1 C2 C3 C4 C5 C6 C7

0 1 0 0 0 0 1

그럼 이 수신기에서는 오류 검출기가 C1, C3, C5, C7을 even parity bit check하여 에러가 발생했음을 인지하여 1을 반환하고, C2, C3, C6, C7을 parity bit check하여 문제가 없으므로 0을 반환하고, C4, C5, C6, C7을 check하여 에러가 있으므로 1을 반환하게 됩니다. 따라서 checker가 반환한 값을 거꾸로 순서대로 적으면 1, 0, 1이 되어 101, 즉 5번째 칸에 문제가 있다고 결론 내리게 되어 C5의 0을 1로 자동으로 수정합니다. 해밍 코드를 이용하면 생성기에서 이런 식으로 데이터를 부호화하여 전송하고, 검출기(검사기)에서 오류를 찾아 스스로 정정하게 됩니다.

1. N bit 비교기에 대해 조사하시오.

5주차 실험에서 이미 1 bit 비교기에 대해 다룬 적 있습니다. 당시 Verilog 코딩을 통해 1 bit 비교기를 구현하였고 그 결과 <Figure 3>와 같은 schematic과 simulation 결과를 통해 <Table 1>의 진리표 얻을 수 있었습니다.



<Figure 3> 1 bit 비교기의 schematic diagram

<Table 1> 1 bit 비교기의 진리표

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | outSame | outDiff | outBig | outSmall |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

이 진리표를 통해 outSame은 두 입력 A와 B가 서로 같을 때이므로 AB+A’B’로 구할 수 있고, outDiff는 두 입력이 서로 다를 때이므로 AB’+A’B로 구할 수 있고, outBig은 AB’을, outSmall은 A’B를 이용하여 구할 수 있다는 사실을 알 수 있었습니다.

위의 예시는 1 bit 비교기의 예시였지만 2 이상의 비트(=n)를 갖는 두 개의 이진수 비교하는 회로를 n bit 비교기라고 합니다. 우선, n이 2일 때, 즉 2비트 비교기에 대해 알아보기 위해 두 개의 이진수 X(X1X2)와 Y(Y1Y2)를 비교한다고 가정할 것입니다. 입력 X1, X2, Y1, Y2에 대한 출력 outSame, outDiff, outBig, outSmall를 나타낸 진리표를 확인해보면 다음 <Table 2>와 같습니다.

<Table 2> 2 bit 비교기의 진리표

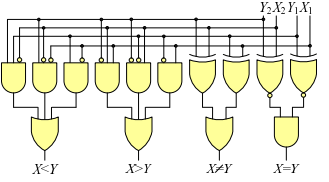
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X1X2(=X) | Y1Y2(=Y) | outSame(=F1) | outDiff(=F2) | outBig(=F3) | outSmall(=F4) |
| 00 | 00 | 1 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 0 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 01 | 00 | 0 | 1 | 1 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 00 | 0 | 1 | 1 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 10 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | 1 |
| 11 | 00 | 0 | 1 | 1 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 10 | 0 | 1 | 1 | 0 |
| 11 | 1 | 0 | 0 | 0 |

텍스트, 낱말맞추기게임이(가) 표시된 사진

자동 생성된 설명따라서, 위의 <Table 2>를 이용해 각 출력에 대한 카르노맵을 작성하고, 각 출력에 대한 논리 함수를 찾으면 다음과 같습니다.

<Figure 4> F1, F2, F3, F4에 대한 카르노맵과 논리 함수

<Figure 4>에서 각 출력에 대한 논리 함수를 구했으므로 이를 이용해 2비트 비교기를 구현해보면 다음 <Figure 5>와 같습니다.



<Figure 5> 2비트 비교기의 schematic diagram

1, 2비트 비교기에 대해서는 위에서 언급한 대로 구현할 수 있고, 4비트 비교기처럼 n이 2보다 커질 때는 4비트 비교기를 여러 개 이용하는 식으로 구현할 수 있습니다. 4비트 비교기는 아래 IC 7485 비교기를 설명하면서 다뤄보도록 하겠습니다.

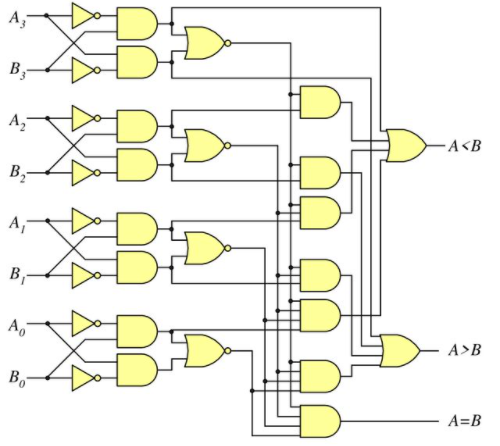
1. IC 7485 비교기에 대해 조사하시오.

IC 7485 비교기는 4비트 비교기로, A3A2A1A0이라는 4비트 입력과 B3B2B1B0이라는 4비트 입력을 비교하여 A가 B보다 클 때는 OA>B를 1로, A가 B보다 작을 때는 OA<B를 1로, A와 B가 서로 같을 때는 OA=B를 1로 출력하는 비교기입니다. 이때, IC 7485 비교기에는 A3~A0와 B3~B0 뿐만 아니라 cascading input이라고 불리는 IA>B, IA<B, IA=B이 추가적인 입력으로 사용되고 cascading input은 하위 자릿수에서 올라오는 입력으로 4비트 비교기를 이용해서 12비트, 16비트 비교기를 구현할 때에 사용하는 입력입니다.

테이블이(가) 표시된 사진

자동 생성된 설명

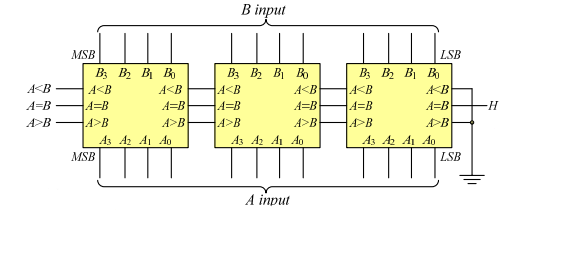
<Figure 6> 4비트 비교기(IC 7485)의 진리표



<Figure 7> 4비트 비교기(IC 7485) 회로

위의 <Figure 6>의 진리표를 이용해서 각 출력에 대한 논리 함수를 표현해보면 outSame은 Ai와 Bi를 XNOR 연산한, 즉 Ai’Bi’+AiBi를 Xi라고 하면 X3, X2, X1, X0을 논리곱(outSame=X3X2X1X0)하여 구할 수 있고, outBig은 outBig=A3B3’+X3A2B2’+X3X2A1B1’+X3X2X1A0B0’으로 구할 수 있으며 outSmall은 A3’B3+X3A2’B2+X3X2A1’B1+X3X2X1X0A0’B0으로 구할 수 있습니다.

12비트 비교기는 이러한 4비트 비교기를 3개 연결하여 다음과 같이 구현할 수 있습니다.



<Figure 8> IC 7485를 사용한 12비트 비교기 회로

1. 기타 이론

디코더란 n개의 비트를 입력으로 받아 2^n개의 출력을 만들 때 사용되는 논리 회로입니다. 이때, 하나의 출력에서 1을 출력하면 다른 출력들은 모두 0을 출력한다는 특징이 있습니다. 만약, 두 개의 입력 A와 B가 있을 때, AB가 00이면 D0에서 1을 출력하고 AB가 01이면 D1이 1, AB가 10이면 D2가 1, AB가 11이면 D3가 1이 됩니다. 이는 8주차에서 다룰 BCD 디코더에서도 다시 나오는 논리 회로이기 때문에 알아두어야 합니다.

디코더의 반대 개념인 인코더라는 논리 회로가 존재합니다. 인코더는 2^n개의 입력을 받아 n개의 출력을 만들 때 사용되는 논리 회로입니다. 입력의 개수보다 출력의 개수가 적기 때문에 암호화한다는 점에서 인코더라고 불리고 반대로 입력의 개수가 출력의 개수보다 적은 디코더는 암호를 해독한다는 점에서 디코더라고 불리는 것입니다. 인코더는 디코더와 정반대로 동작하는데, 입력 2^n개 중 단 하나만이 1이고 나머지는 0이어야 하며 입력 A3A2A1A0이 0001일 때는 출력 AB가 00이고 A3A2A1A0이 0010이면 AB가 01, A3A2A1A0이 0100이면 AB가 10, A3A2A1A0이 1000이면 AB는 11이 됩니다.