7주차 결과보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20211547 이름: 신지원

**1.**

**- Even Parity bit generator**

Parity bit generator 는 에러를 검출하는 비트를 생성하는 생성기로 이번 실습에서는 짝수 Parity bit generator 에 대한 코드를 직접 짜보았다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위는 Even parity bit generator 의 design 코드다. Input 인 a,b,c,d 의 1-bit 수가 짝수면 0으로 나타내야 하기 때문에 a,b,c,d 를 모두 XOR gate 로 도출하였다. 만약 홀수 개일 때 1이 pec 의 결과로 나타날 수 있도록 코드를 작성하였다.

스크린샷, 디스플레이, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

코드의 결과로 위와 같은 simulation 이 도출되었다. a,b,c,d 순으로 (1,0,0,0) 일 때 pec 의 값이 1이며, (1,1,1,1) 일 때 pec의 값이 0인 것으로 보아 even parity bit generator 의 역할을 잘 해내고 있음을 알 수 있다. 코드와 시뮬레이션을 기반으로 아래 진리표를 작성하였다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| In aa | In bb | In cc | In dd | Out pec |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd ab | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 0 | 1 |
| 01 | 1 | 0 | 1 | 0 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |

F = a⊕b⊕c⊕d

위는 진리표를 바탕으로 그린 카르노맵이다. XOR gate 를 사용하였기 때문에 짐작해볼 수 있었겠지만 카르노맵은 두 개 이상 묶이지 않고 oxox/xoxo 형식으로 위치해 있는 것을 볼 수 있다. 특이한 형태로 한 눈에 SOP 식을 완성할 수 있었다. 이는 실습에서 구현한 코드와 같다.

**- Even Parity bit checker**

Parity bit Checker는 비트를 통해 에러를 검출하는 검사기다. Even Parity bit checker 는 비트를 모두 더한 값이 짝수(0) 이라면 정상적으로 진행되어 0을 도출하고, 홀수(1)이라면 에러 값인 1을 도출한다. 이번 실습에서는 Even parity bit Checker에 대한 코드를 직접 짜보았다

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위는 Even parity bit checker 의 design 코드다. Input 인 a,b,c,d,p 중 1-bit 수가 짝수 개면 0으로 나타내야 하기 때문에 a,b,c,d,p 를 모두 XOR gate 로 도출하였다. 만약 홀수 개일 때 1이 pec 의 결과로 나타날 수 있도록 코드를 작성하였다.

스크린샷, 도표, 디스플레이, 텍스트이(가) 표시된 사진

자동 생성된 설명

코드의 결과로 위와 같은 simulation 이 도출되었다. a,b,c,d,p 순으로 (1,0,0,0,0) 일 때 pec 의 값이 1이며, (1,1,0,1,1) 일 때 pec의 값이 0인 것으로 보아 even parity bit checker 의 역할을 잘 해내고 있음을 알 수 있다. 코드와 시뮬레이션을 기반으로 아래 진리표를 작성하였다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| In aa | In bb | In cc | In dd | In pp | Out pec |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

a = 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| dp bc | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 0 | 1 |
| 01 | 1 | 0 | 1 | 0 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |

a = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| dp bc | 00 | 01 | 11 | 10 |
| 00 | 1 | 0 | 1 | 0 |
| 01 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 |

F = a⊕b⊕c⊕d⊕p

위는 진리표를 바탕으로 그린 카르노맵이다. Input 이 4개를 넘기 때문에 a를 기준으로 나누어 카르노맵을 작성하였으며, XOR gate 를 사용하였기 때문에 짐작해볼 수 있었겠지만 카르노맵은 두 개 이상 묶이지 않고 oxox/xoxo 형식으로 위치해 있는 것을 볼 수 있다. 특이한 형태로 한 눈에 SOP 식을 완성할 수 있었다. 이는 실습에서 구현한 코드와 같다.

**2.**

**- Odd Parity bit generator**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이번 실습에서는 짝수 Parity bit generator 에 대한 코드만 다루었으며 홀수 Parity bit gerator 는 실습을 응용하여 코드를 구현하였다. Even parity bit generator 에서 not 만 제거하여 a,b,c,d 의 XOR gate 한 결과값과 같도록 작성하였다. 따라서 Input 인 a,b,c,d 의 1-bit 수가 짝수면 1으로 나타내야 하기 때문에 a,b,c,d 를 모두 XOR gate 로 도출하여 not gate 를 통과한 값으로 선언하였다. 만약 홀수 개일 때 0이 pec 의 결과로 나타날 수 있도록 코드를 작성하였다.

스크린샷, 디스플레이, 텍스트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

코드의 결과로 위와 같은 simulation 이 도출되었다. a,b,c,d 순으로 (1,0,0,0) 일 때 pec 의 값이 0이며, (1,1,1,1) 일 때 pec의 값이 1인 것으로 보아 odd parity bit generator 의 역할을 잘 해내고 있음을 알 수 있다. 코드와 시뮬레이션을 기반으로 아래 진리표를 작성하였다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| In aa | In bb | In cc | In dd | Out pec |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd ab | 00 | 01 | 11 | 10 |
| 00 | 1 | 0 | 1 | 0 |
| 01 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 |

F = ~(a⊕b⊕c⊕d)

위는 진리표를 바탕으로 그린 카르노맵이다. Even parity bit generator 와 마찬가지로 XOR gate 를 사용하였기 때문에 짐작해볼 수 있었겠지만 카르노맵은 두 개 이상 묶이지 않고 xoxo/oxox 형식으로 위치해 있는 것을 볼 수 있다. 카르노맵으로 도출한 SOP 식은 design 코드와 같다.

**- Odd Parity bit checker**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

실습에서는 짝수 Parity bit checker 에 대한 코드만 다루었으며 홀수 Parity bit checker 는 실습을 응용하여 코드를 구현하였다. Even parity bit checker 에서 not 만 제거하여 a,b,c,d,p 의 XOR gate 한 결과값과 같도록 작성하였다. 따라서 Input 인 a,b,c,d,p 의 1-bit 수가 홀수면 0으로 나타내야 하기 때문에 a,b,c,d,p 를 모두 XOR gate 로 도출하여 not gate 를 통과한 값으로 선언하였다. 만약 짝수 개일 때 1이 pec 의 결과로 나타날 수 있도록 코드를 작성하였다.

스크린샷, 소프트웨어, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

코드의 결과로 위와 같은 simulation 이 도출되었다. a,b,c,d,p 순으로 (1,0,0,0,0) 일 때 pec 의 값이 0이며, (1,1,0,1,1) 일 때 pec의 값이 1인 것으로 보아 odd parity bit checker 의 역할을 잘 해내고 있음을 알 수 있다. 코드와 시뮬레이션을 기반으로 아래 진리표를 작성하였다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| In aa | In bb | In cc | In dd | In pp | Out pec |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

a = 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| dp bc | 00 | 01 | 11 | 10 |
| 00 | 1 | 0 | 1 | 0 |
| 01 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 |

a = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| dp bc | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 0 | 1 |
| 01 | 1 | 0 | 1 | 0 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |

F = ~(a⊕b⊕c⊕d⊕p)

위는 진리표를 바탕으로 그린 카르노맵이다. Input 이 4개를 넘기 때문에 a를 기준으로 나누어 카르노맵을 작성하였으며, XOR gate 를 사용하였기 때문에 짐작해볼 수 있었겠지만 카르노맵은 두 개 이상 묶이지 않고 xoxo/oxox 형식으로 위치해 있는 것을 볼 수 있다. 카르노맵으로 도출한 SOP 식은 design 코드와 같다.

**3.**

2-bit 비교기란, 2자릿수를 가진 2진수의 값을 비교하는 비교기다. 비교순서는 상위자릿수에서 하위자릿수로 이어진다. 코드를 구현할 때 앞선 실습처럼 직관적으로 코드를 작성하기엔 어려움이 있어 먼저 카르노맵을 그린 뒤 코드를 구현하였다. f1은 a>b, f2 는 a=b, f3 은 a<b 일 때를 카르노맵으로 나타내어 SOP 식을 도출하였다.

Output f1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd ab | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 1 | 1 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 |

Output f2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| dp bc | 00 | 01 | 11 | 10 |
| 00 | 1 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 |
| 10 | 0 | 0 | 0 | 1 |

Output f3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| dp bc | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 11 | 1 | 1 | 0 | 1 |
| 10 | 1 | 1 | 0 | 0 |

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위처럼 코드를 구현하였으며 시뮬레이션은 아래와 같았다.

스크린샷, 디스플레이, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

(a,b) 와 (c,d) 를 비교하여 f1, f2, f3의 각각의 값이 전자가 더 클 때, 같을 때, 후자가 더 클 때를 도출하는지 확인해보았다. 가장 앞부분에서 (1,0) 과 (0,0) 일 때 f1에만 1의 값이 나타나고, (1,1) 과 (1,1) 일 때 f2에만 1의 값이 나타나며, (1,0) 과 (1,1) 일 때 f3에만 1의 값이 나타나기 때문에 시뮬레이션 상에서 위 코드는 적절하게 구현하였음을 알 수 있다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In aa | In bb | In cc | In dd | Out f1 | Out f2 | Out f3 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

위는 코드와 시뮬레이션을 바탕으로 진리표다.

**4.**

 even parity bit generator의 논리식은 a⊕b⊕c⊕d 이고, odd parity bit generator의 논리식은 ~(a⊕b⊕c⊕d) 이었다. checker 에 대한 실습에서는 even parity bit checker 의 논리식이 a⊕b⊕c⊕d⊕f 이고, odd parity bit checker 의 논리식이 ~( a⊕b⊕c⊕d⊕f ) 이다. 따라서 생성기와 확인기 모두 짝수와 홀수의 논리식이 보수 관계임을 알 수 있었다. 2-bit 를 사용하여 이를 비교하는 실습에서는, A > B일 때 F1, A = b일 때 F2, A > B일 때 F3 의 값이 각각 1이 됨을 고려하여 보고서를 작성하였다.

**5.**

검사기에는 실습에서 사용해본 parity bit checker, n-bit checker 뿐만 아니라 블록 합 검사 또한 존재한다. 합 겁사는 짝수나 홀수를 사용하지 않고, 블록에 수평, 수직 parity 를 사용하는 방식이다. 이런 방식으로는 colum 단위의 오류를 검사할 수 있다. 우리가 실습한 짝수 홀수에 비해 홀짝에 구애를 받진 않는다는 장점이 있지만 많은 오버헤드가 발생할 수 있다는 단점이 있다.