7주차 결과보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20212022 이름: 이예준

**1.**

**-Even Parity bit generator**

스크린샷, 디스플레이, 소프트웨어, 텍스트이(가) 표시된 사진

자동 생성된 설명

\* Even Parity bit generator simulation결과

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **In A** | **In B** | **In C** | **In D** | **Out P** |
| 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 | 0 |

\* Even Parity bit generator 진리표

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

\* Even Parity bit generator K-map

Even Parity bit generator는 전달되는 정보의 bit의 개수를 짝수로 맞춰 주기 위해

Bit의 1의 개수가 짝수이면 0, 홀수이면 1을 반환하여 parity bit를 생성해준다.

위의 진리표를 통해 K-map을 만들어주고, 그 K-map을 통해 generator의 논리식을 만들면,

이 된다.

이때, 을 이용해 논리식을 간소화시키면,

이 된다.

**-Even Parity bit checker**

스크린샷, 텍스트, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

\* Even Parity bit checker simulation결과

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **In A** | **In B** | **In C** | **In D** | **In P** | **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 |

\* Even Parity bit checker 진리표

(a = 1) (a = 0)

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

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

\* Even Parity bit checker K-map\_1 \* Even Parity bit checker K-map\_2

Even Parity bit checker는 4개의 입력값(a,b,c,d)과 1개의 parity bit(p)를 입력 받아 1의 개수가 짝수이면 오류가 없다는 뜻으로 0, 홀수이면 오류가 있다는 뜻으로 1을 반환한다.

위의 진리표를 통해 K-map을 만들어주고, 그 K-map을 통해 checker의 논리식을 만들면,

이 된다.

이때, generator와 마찬가지로 을 이용해 논리식을 간소화시키면,

이 된다.

**2.**

**- Odd Parity bit generator**

소프트웨어, 멀티미디어 소프트웨어, 스크린샷이(가) 표시된 사진

자동 생성된 설명

\* Odd Parity bit generator simulation결과

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **In A** | **In B** | **In C** | **In D** | **Out P** |
| 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 |

\* Odd Parity bit generator 진리표

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

\* Odd Parity bit generator K-map

Odd Parity bit generator는 전달되는 정보의 bit의 개수를 홀수로 맞춰 주기 위해

Bit의 1의 개수가 홀수이면 0, 짝수이면 1을 반환하여 parity bit를 생성해준다.

위의 진리표를 통해 K-map을 만들어주고, 그 K-map을 통해 generator의 논리식을 만들면,

이 된다.

이때, 을 이용해 논리식을 간소화시키면,

이 된다.

**- Odd Parity bit checker**

스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\* Odd Parity bit checker simulation결과

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **In A** | **In B** | **In C** | **In D** | **In P** | **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 |

\* Odd Parity bit checker 진리표

(a = 0) (a = 1)

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

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

\* Odd Parity bit checker K-map\_1 \* Odd Parity bit checker K-map\_2

Odd Parity bit checker는 4개의 입력값(a,b,c,d)과 1개의 parity bit(p)를 입력 받아 1의 개수가 홀수이면 오류가 없다는 뜻으로 0, 짝수이면 오류가 있다는 뜻으로 1을 반환한다.

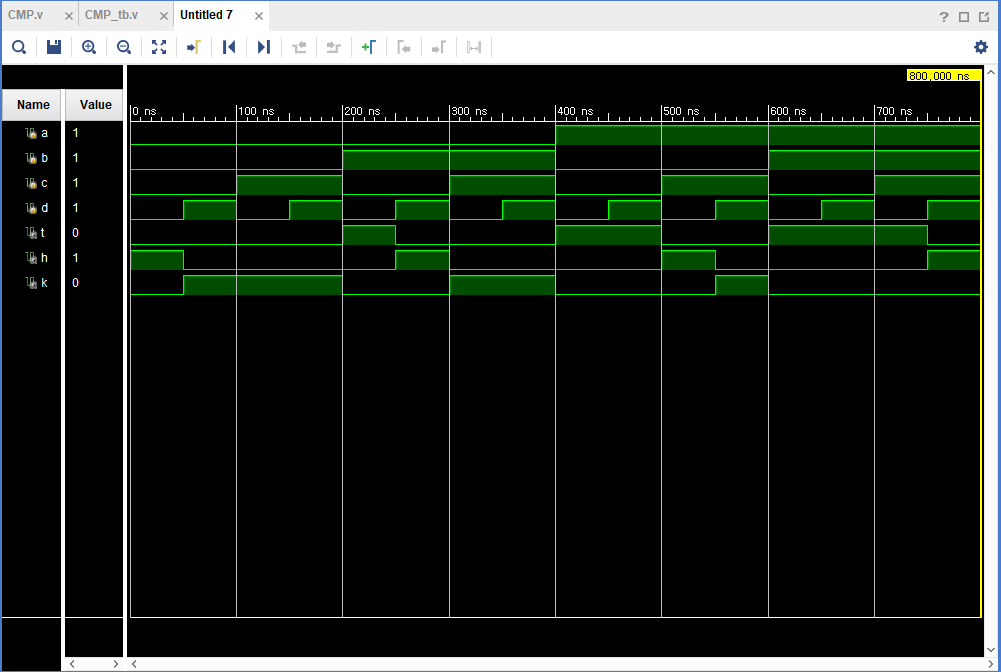
위의 진리표를 통해 K-map을 만들어주고, 그 K-map을 통해 checker의 논리식을 만들면,

이 된다.

이때, generator와 마찬가지로 을 이용해 논리식을 간소화시키면,

이 된다.

**3.**

**- 2-bit binary comparator** 

\*2-bit binary comparator simulation 결과

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **In A** | **In B** | **In C** | **In D** | **Out T** | **Out H** | **Out K** |
| 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 |

\*2-bit binary comparator 진리표

(T)

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

간소화 →

(H)

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

간소화 →

(K)

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

간소화 →

4입력값(a,b,c,d)을 받는데 a와 b, c와 d를 짝지어서 2-bit를 비교한다.

T는 ab 2-bit가 cd 2-bit보다 더 클 때 1을 반환하고, 그 외에는 0을 반환한다.

H는 ab 2-bit와 cd 2-bit가 서로 같을 때, 1을 반환하고, 그 외에는 0을 반환한다,

K는 cd 2-bit가 ab 2-bit보다 더 클 때 1을 반환하고, 그 외에는 0을 반환한다.

**4.**

실험에서 Even Parity bit generator와 checker가 1의 개수가 짝수일 때는 0, 홀수일 때는 1을 반환하고, Odd Parity bit generator와 checker는 1의 개수가 짝수일 때는 1, 홀수일 때는 0을 반환하는 것을 Simulation결과와 FPGA 보드 구현 결과를 통해 확인할 수 있었다.

generator와 checker 모두 카르노 맵을 통해 AND gate와 OR gate로 구현할 수 있었고,

그 논리식을 XOR gate로 간소화할 수 있다는 것까지 확인했다.

논의 사항으로 간소화하는 방법에 대해 말하고자 하는데, 이번 실험에서 단순히 카르노맵을 통해 식을 구현하다 보니 굉장히 긴 논리식으로 Verilog코드를 짜게 되었다.

하지만 보고서 작성을 하면서 알아본 결과, 아주 간단하게 XOR gate로 간소화할 수 있는 것을 알면서 카르노 맵 뿐만 아니라 Boolean대수 정리도 함께 적용하여 간소화방법을 다각화할 필요성을 느꼈다.

**5.**

Parity bit를 통해서 오류를 찾아내는 방법도 있지만 Checksum이라는 방법을 통해

수신된 데이터에 오류가 없는지 검사하는 방법도 있다.

예를 들어 2진수의 4bit 데이터 (1011), (1100), (0011)가 있을 때:

1. 모든 bit를 더하면 (11010)이 된다.

2. 캐리를 버려 (1010)을 만든다.

3. 2’s Complement를 취해 (0110)을 얻으면, 이것이 Checksum bit이다.

데이터에 Checksum을 붙여 송신하고 수신자는 받아서 체크한다.

1. 데이터의 모든 bit와 Checksum bit를 더하면 (100000)이 된다.

2. 캐리를 버리면 (0000)이 되는데, 이처럼 오류가 없으면 (0000)이 나오고

다른 값이 나오면 데이터가 깨졌다고 판별한다.