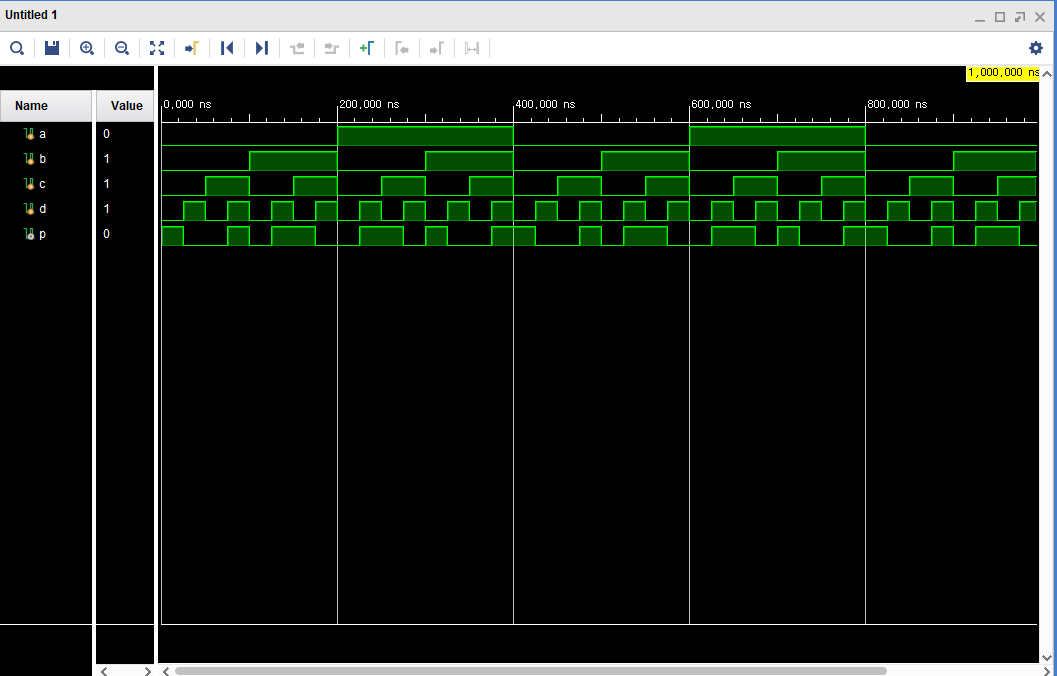
7주차 결과보고서

전공 : 컴퓨터공학과 학년 : 2학년 학번 : 20201597 이름 : 신동준

1. **Even Parity bit generator 및 checker 의 simulation 결과 및 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)**

**<generator>**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |



텍스트, 화이트보드이(가) 표시된 사진

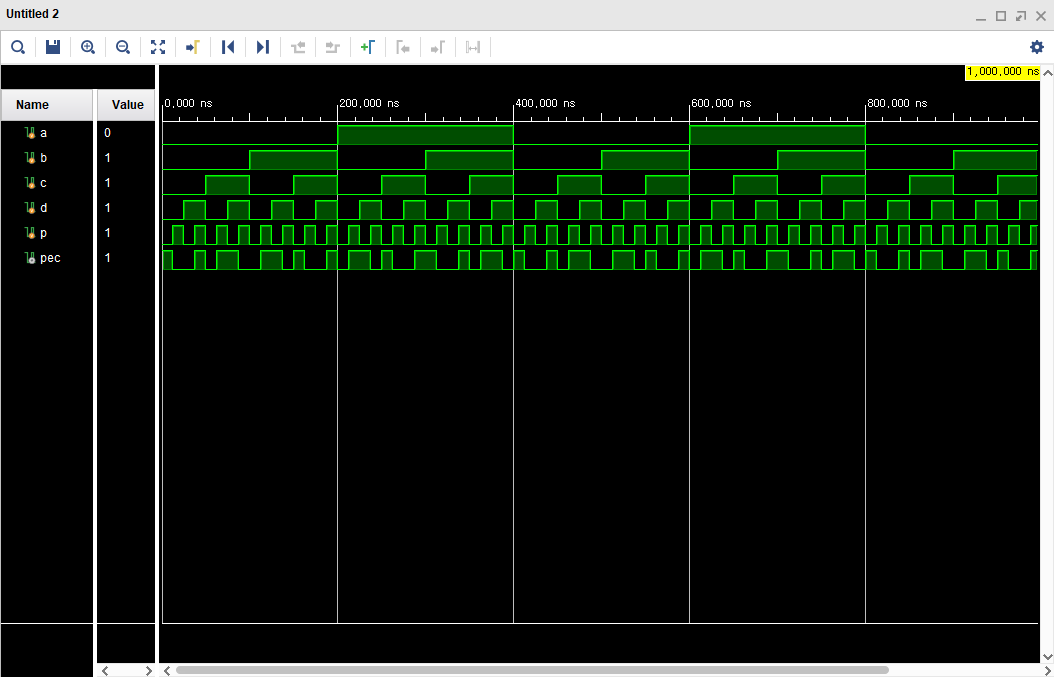
자동 생성된 설명

Even Parity bit Generator는 XOR 연산으로 묶어서 판별할 수 있다. Input 받은 bit에서 1의 개수가 짝수면 parity bit으로 1을 붙이고, 1의 개수가 홀수면 0을 붙인다.

[p = ~((a ^ b) ^ (c ^ d))]

**<checker>**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |



텍스트이(가) 표시된 사진

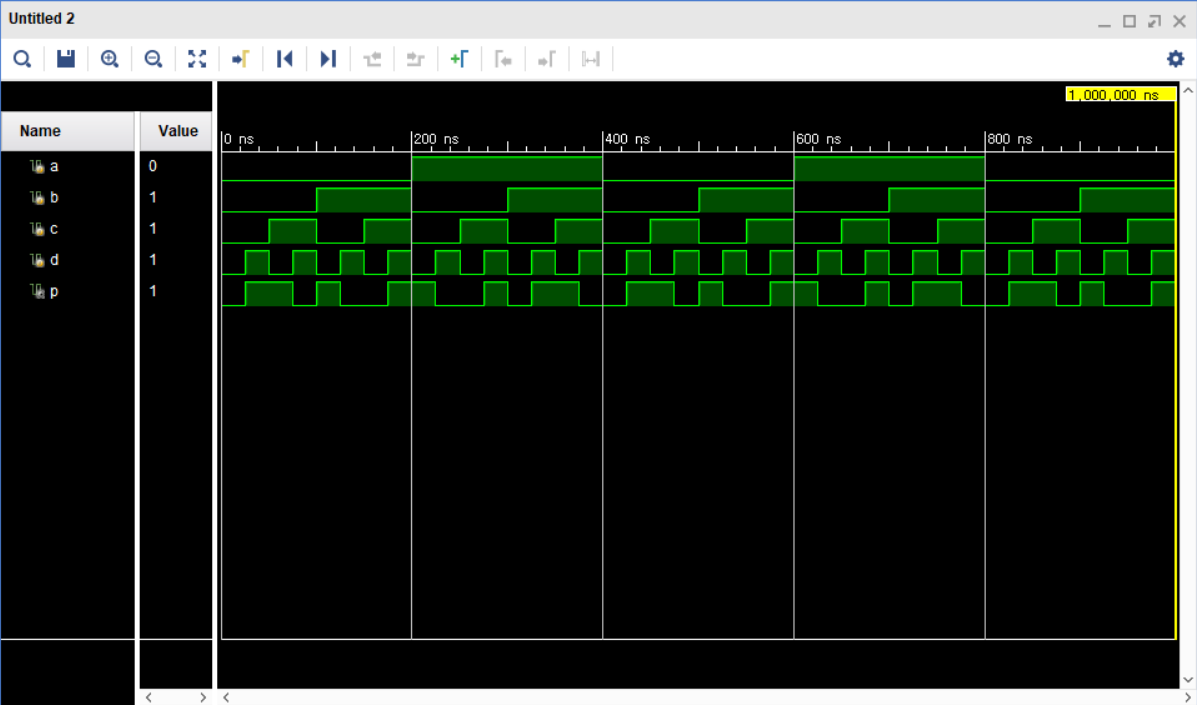
자동 생성된 설명

Even Parity Bit Checker는 parity bit까지 총 5개의 bit input 중 참 거짓을 판별한다. 들어온 bit 개수가 홀수면 정상이기 때문에 위의 Generating과 동일하게 XOR 연산을 이용해서 1의 개수를 판별해준다. [PEC = ~((a ^ b) ^ ((c ^ d) ^ p))]

1. **Odd Parity bit generator 및 checker 의 simulation 결과 및 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)**

**<generator>**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |



텍스트, 전자기기, 검은색, 키보드이(가) 표시된 사진

자동 생성된 설명

Odd Parity bit Generator는 XOR 연산으로 묶어서 판별할 수 있다. Input 받은 bit에서 1의 개수가 짝수면 parity bit으로 0을 붙이고, 1의 개수가 홀수면 1을 붙인다.

[p = ((a ^ b) ^ (c ^ d))]

**<Checker>**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

Odd Parity Bit Checker는 parity bit까지 총 5개의 bit input 중 참 거짓을 판별한다. 들어온 bit 개수가 짝수면 정상이기 때문에 위의 Generating과 동일하게 XOR 연산을 이용해서 1의 개수를 판별해준다. [PEC = ((a ^ b) ^ ((c ^ d) ^ p))]

1. **2-bit binary comparator simulation 결과 및 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In A | In B | In C | In D | 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 |



텍스트, 옅은이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

두 숫자를 비교하는 연산에서 두 숫자가 같은 경우는 직관적이다. 같은 경우에 true 아닌경우 false를 출력하고 k-map 상에서도 각각의 경우가 (2^1, 2^0 위치)가 같을 때를 and 연산으로 묶어서 다음과 같은 형태로 나타낼 수 있다.

[~(a1 ^ b1) & ~(a2 ^ b2)]

두 숫자가 다른 경우 한쪽이 큰 경우에 카르노 맵을 기준으로 SOP 방식으로 연산자를 나열해보면 2^1부터 먼저 비교해보고 아닌 경우 2^0을 비교해보는 방식을 볼 수 있다.

AB > CD인 경우는 아래와 같고 AB < CD는 여기에 대각행렬이 아닌 곳에 not을 씌운 형태이다.

[(a1 & ~b1) | (a2 & ~b1 & ~b2) | (a1 & a2 & ~b2)]

1. **결과 검토 및 논의 사항.**

두 종류의 Parity bit generator / checker를 만들어 보았는데, 이들은 전송된 데이터의 품질을 보증하고, 확인하는 역할을 하게 된다. 송신자와 수신자는 어떤 종류의 parity bit을 사용할 것인지 사전에 협의가 필요하다.

Odd를 기준으로 설명하면(even의 경우 결과가 반대이다) 최초의 송신하고자 하는 bit data의 1 개수를 파악하고 홀수개라면 parity bit으로 1을 덧붙여준다.(generator 역할)

Checker는 parity bit을 포함한 bit의 개수가 짝수라면 정상임으로 1을 반환한다.

2-bit comparator는 2비트의 두 수 중 어떤 쪽이 큰지 비교기이다. 같을 때 전자가 클 때, 후자가 클 때의 총 3개의 출력 값으로 구성된다. 4-bit k-map을 그려서 각각의 상황을 가장 간단한 식으로 나타내고 그를 바탕으로 wave form을 그려보니 이론과 동일하게 나오는 모습을 볼 수 있었다.

1. **추가 이론 조사 및 작성.**

Parity bit을 이용한 우리가 구현한 방법은 사실상 XOR 연산만으로 구현할 수 있고, 직관적이게 전송된 정보의 변조 여부를 확인할 수 있다. 하지만 정보의 길이가 길어지고, 부당한 목적을 가진 사람이 parity bit까지 고려해서 변조를 할 수도 있다.

이를 대비해서 순환 중복검사 CRC 등이 사용되기도 하고, parity bit을 여러 개 사용하는 해밍 ECC 등의 방법 들이 존재한다.