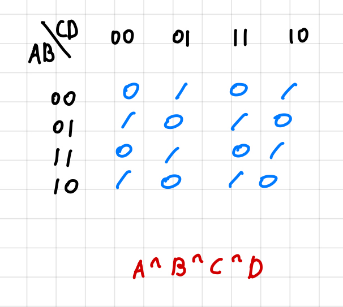
7주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231609 이름: 정희선

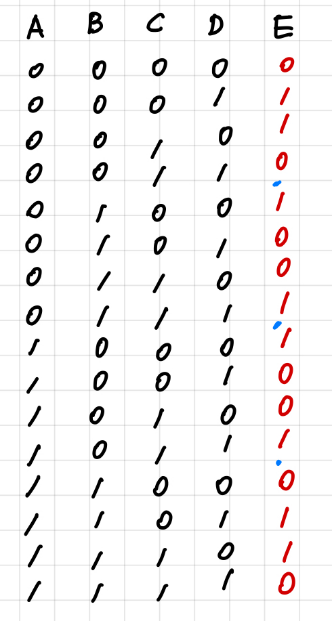
**1.**

**1.1) Even Parity bit generator**

**1) K-map**

****

**2) Truth table**

****

**3) Verilog**

*`timescale 1ns / 1ps*

*module boolean(*

*input a,*

*input b,*

*input c,*

*input d,*

*output p*

*);*

*assign p = a^b^c^d;*

*endmodule*

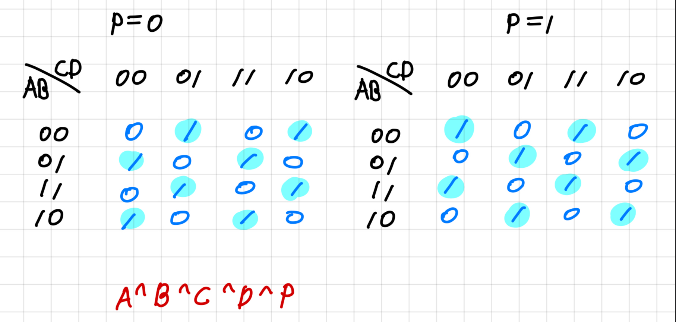
**4) simulation**

**스크린샷, 라인, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

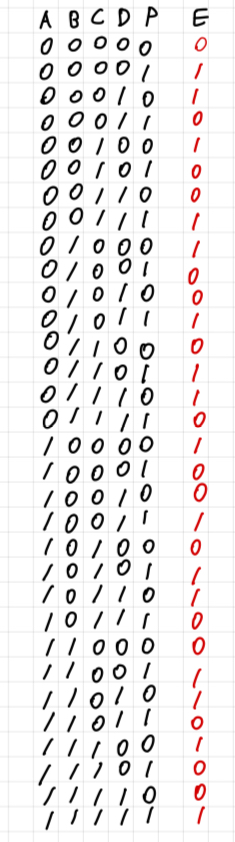
자동 생성된 설명**

**1.2) Even Parity bit checker**

**1) K-map**

****

**2) Truth table**

****

**3) Verilog**

*`timescale 1ns / 1ps*

*module boolean(*

*input a,*

*input b,*

*input c,*

*input d,*

*input p,*

*output e*

*);*

*assign e = a^b^c^d^p;*

*endmodule*

**4) simulation**

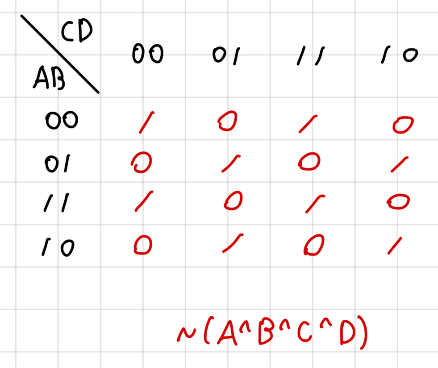
**스크린샷, 멀티미디어 소프트웨어, 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

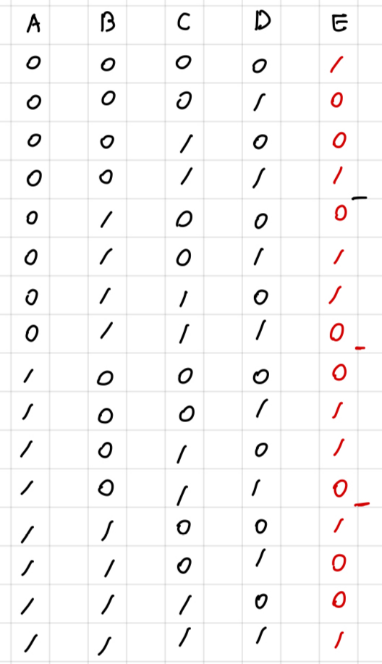
**2.**

**2.1) Odd Parity bit generator**

**1) K-map**

****

**2) Truth table**

****

**3) Verilog**

*`timescale 1ns / 1ps*

*module boolean(*

*input a,*

*input b,*

*input c,*

*input d,*

*output p*

*);*

*assign p = ~(a^b^c^d);*

*endmodule*

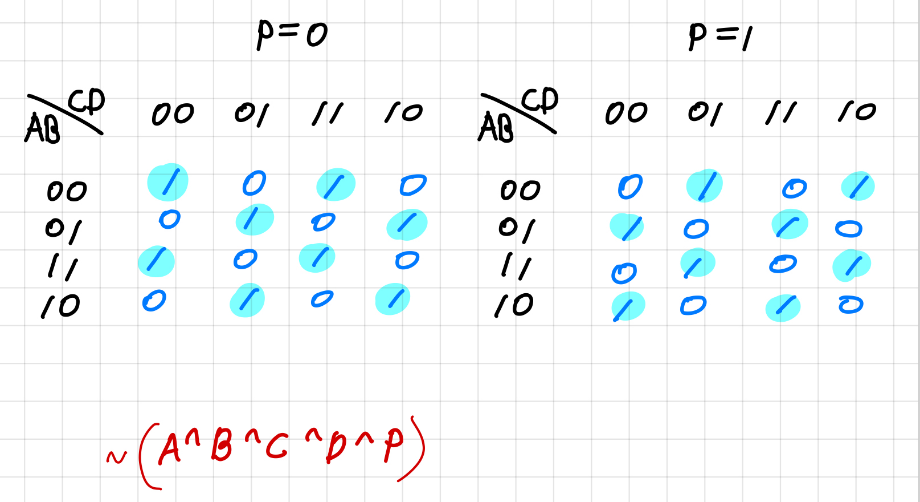
**4) simulation**

**스크린샷, 텍스트, 소프트웨어, 번호이(가) 표시된 사진

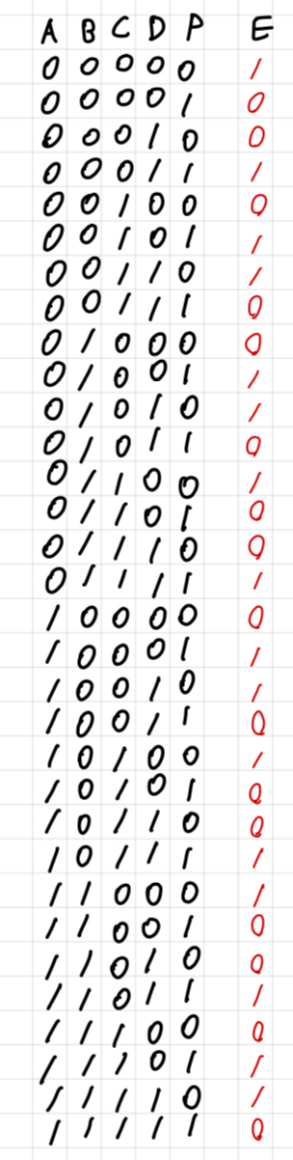
자동 생성된 설명**

**2.2) Odd Parity bit checker**

**1) K-map**

****

**2) Truth table**

****

**3) Verilog**

*`timescale 1ns / 1ps*

*module boolean(*

*input a,*

*input b,*

*input c,*

*input d,*

*input p,*

*output e*

*);*

*assign e = ~(a^b^c^d^p);*

*endmodule*

**4) simulation**

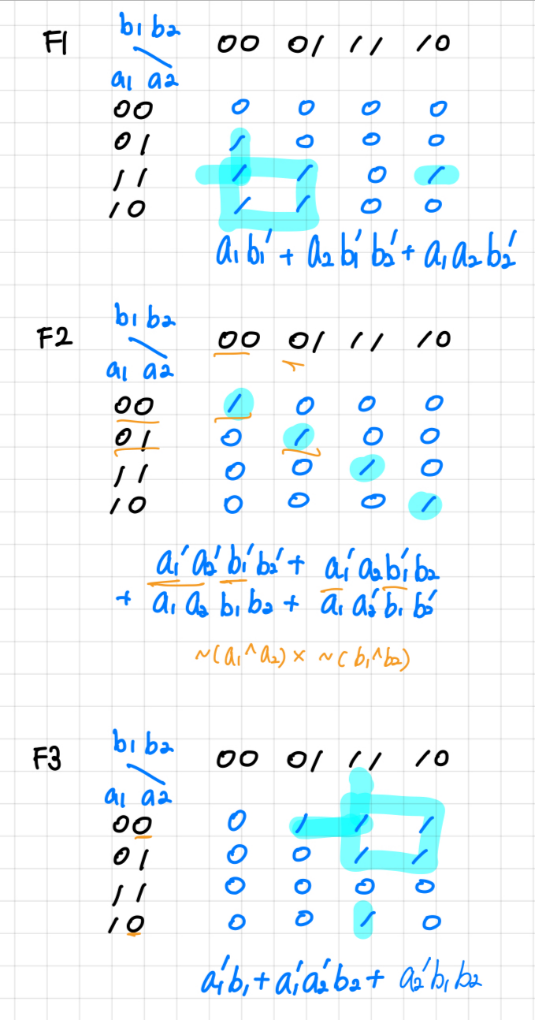
스크린샷, 소프트웨어, 멀티미디어 소프트웨어, 텍스트이(가) 표시된 사진

자동 생성된 설명

**3.**

**2-bit binary comparator**

1. **K-map**

****

1. **Truth table**

****

1. **Verilog**

*timescale 1ns / 1ps*

*module boolean(*

*input a1,*

*input a2,*

*input b1,*

*input b2,*

*output f1,*

*output f2,*

*output f3*

*);*

*assign f1 = (a1&(~b1)) | (a2&(~b1)&(~b2)) | (a1&a2&(~b2));*

*assign f2 = ((~a1)&(~a2)&(~b1)&(~b2)) | ((~a1)&a2&(~b1)&b2) | (a1&(~a2)&b1&(~b2)) | (a1&a2&b1&b2);*

*assign f3 = ((~a1)&b1) | (~a1)&(~a2)&b2 | ((~a2)&b1&b2);*

*endmodule*

**4) simulation**

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

자동 생성된 설명

**4.**

**Even parity bit generator/checker, odd parity bit generator/checker**는 데이터 전송에서 전송 오류를 감지하기 위해 사용된다. Even parity는 전체 비트의 1의 개수가 짝수가 되도록 만들었다. 구체적으로 generator는 입력 비트들의 전체 1의 개수가 짝수가 되도록, checker는 입력 비트와 패리티 비트를 합한 전체 1의 개수가 짝수인지 확인했다. 마찬가지로 odd parity는 1의 개수가 홀수가 되도록 만든다. Generator는 입력 비트들의 전체 1의 개수가 홀수가 되도록, checker는 입력 비트와 패리티 비트를 합한 전체 1의 개수가 홀수인지 확인한다. 두 경우 모두 전송 중 한 비트가 잘못되어 짝수(홀수)개의 1이 검출된다면 checker가 오류를 감지해 비정상으로 판별한다.

**2 bit binary comparator**는 두개의 2 bit 숫자를 비교하여 A>B, A=B, A<B의 상태를 판별한다. 모든 가능한 입력 조합에서 비교가 정확히 이루어지고 있는 것을 확인했다. 이 때 각 경우에 대해 하나의 출력만 활성화되도록 해야 한다.

진리표를 기반으로 k-map을 작성하고, 이를 통해 **논리식을 간소화**함으로써 회로의 복잡도를 줄일 수 있었다. 회로의 규모가 커질 때 k-map을 통해 더 적은 게이트를 사용해 동일한 기능을 수행하는 회로를 설계할 수 있다는 것을 알 수 있다.

Parity generator, checker는 오류 검출 기능만을 수행할 수 있다. 이를 보완하여 **오류 수정 기능**을 추가하거나 더 큰 비트 길이에 대해 확장하는 방법을 고려할 수 있다. 예를 들어 hamming code와 같은 기능을 추가하면 오류를 수정할 수 있는 기능을 더해 기존 parity check의 단점을 보완할 수 있다.

**5.**

**4-bit comparator**: 두 개의 4비트 값을 비교한다. 비교 연산을 수행하는 다단 구조로 이루어져 있다.

**확장형 n-bit comparator**: 4비트 이상의 비교가 필요한 경우 사용되는 comparator이다. 여러 4 bit comparator를 병렬로 연결하여 비교 비트를 확장한다. 또는 cascade 연결 방식을 통해 비트를 여러 단계로 연결하여 더 큰 n 비트 비교를 수행한다.

**병렬-직렬 비교 방식**: 이는 비교 속도를 높이기 위해 설계된 고속 비교 방식이다. 특히 데이터 비트가 매우 많거나 빠른 비교가 요구되는 경우 사용된다. 상위 비트를 병렬로 비교한 뒤, 필요한 경우에만 하위 비트를 직렬로 비교하는 방식으로 속도를 높인다.