7주차 결과보고서

20191621 이민영

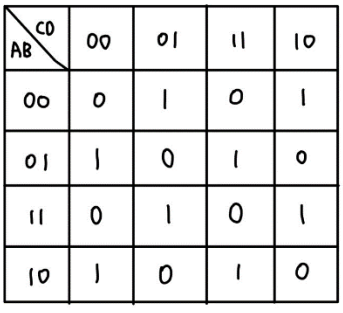
1. Even Parity Bit generator 및 checker의 simulation 결과 및 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)

1) Even Parity Bit generator

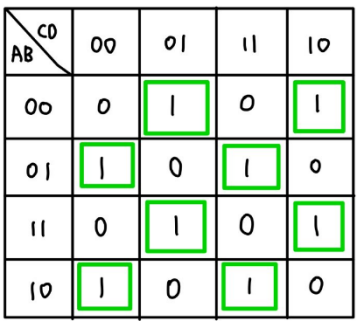
Input A	Input B	Input C	Input D	Output E
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Even Parity Bit의 경우, 1의 개수가 짝수개로 이루어지도록 Parity Bit을 설정해주어야 한다. 즉 Input의 1의 개수가 홀수 개일 경우, Output은 1이 되고, Input의 1의 개수가 짝수일 경우 Output은 0이 되어야 한다.

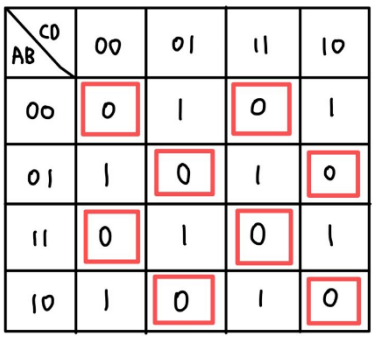
<k-map>



<카르노 맵>



<SOP>



<POS>

① SOP

1로 묶인 항들을 적어보면

A′B′C′D + A′B′CD′ + A′BC′D′ + A′BCD + ABC′D + ABCD′ + AB′C′D′ + AB′CD 이다.

= A′B′(C′D + CD′) + A′B(C′D′+CD) + AB(C′D+CD′) + AB′(C′D′+CD)

=A′B′(C⊕D) + (C′D′+CD)(A′B+AB′) + AB(C⊕D)

= (C⊕D)(A′B′+AB) + (C′D′+CD)(A⊕B)

= (C⊕D)(A⊕B)′ + (C⊕D)′(A⊕B)

=(C⊕D⊕A⊕B) = A⊕B⊕C⊕D 이다.

② POS

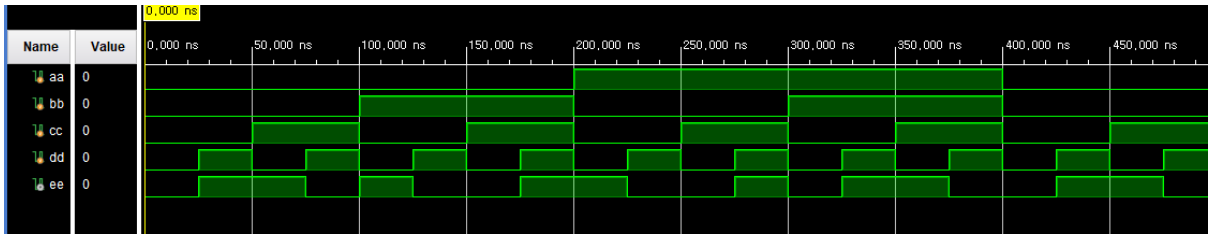
0으로 묶인 항들을 적어보면

보수 = A′B′C′D′ + A′B′CD + A′BC′D + A′BCD′ + ABC′D′ + ABCD + AB′C′D + AB′CD′ 이므로 다시 보수를 취해주면,

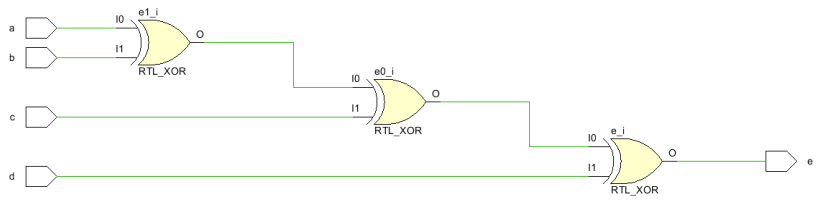
(A′B′C′D′ + A′B′CD + A′BC′D + A′BCD′ + ABC′D′ + ABCD + AB′C′D + AB′CD′)′

= (A′B′C′D′)′(A′B′CD)′(A′BC′D)′(A′BCD′)′(ABC′D′)′(ABCD)′(AB′C′D)′(AB′CD′)′

=(A+B+C+D)(A+B+C′+D′)(A+B′+C+D′)(A+B′+C′+D)(A′+B′+C+D)(A′+B′+C′+D′)(A′+B+C+D′)(A′+B+C′+D) 이다.



<Simulation 결과>



<Schematic>

Even Parity Bit Generator의 진리표를 작성하고 진리표를 기반으로 카르노 맵을 작성하였다. 카르노 맵에서 1로 묶어서 표현한 SOP 방법과 0으로 묶어서 보수를 취하는 방식으로 표현한 POS 방법으로 식을 표현하였다. 이 때, SOP로 나타낸 식은 식의 정리를 통해서 input 끼리의 XOR 값인 A⊕B⊕C⊕D이 된다는 것을 알 수 있었다.

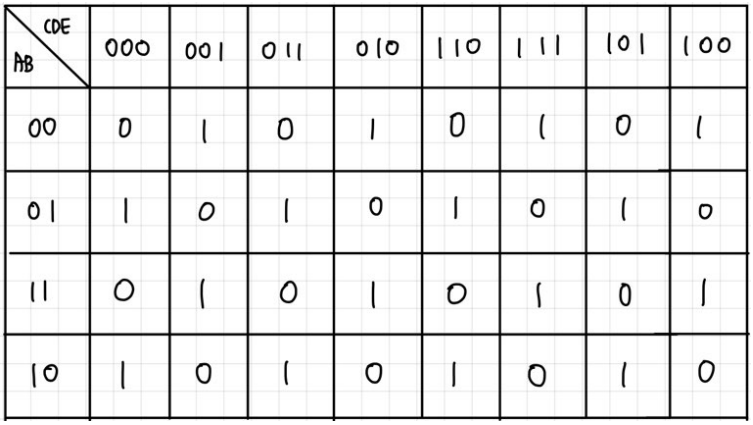
Simulation의 결과를 통해서 진리표와 동일한 결과가 출력된다는 것을 확인할 수 있었고, Schematic을 통해서도 확인해볼 수 있었다.

2) Even Parity Bit Checker

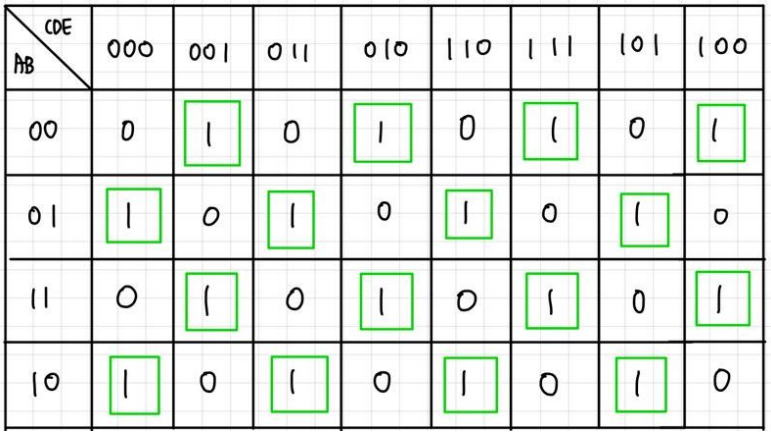
Input A	Input B	Input C	Input D	Input E	Output PEC
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	1
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	0	1	1	1
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	0	0	1	1	1
1	0	1	0	1	1
1	1	0	0	1	1
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	0	0	1

1	1	1	1	0	0
1	1	1	0	1	0
1	1	0	1	1	0
1	0	1	1	1	0
1	1	1	1	1	1

Even Parity bit checker는 정상적으로 Even parity bit이 설정되어서 1의 개수가 짝수일 때는 Output이 0으로 설정되고 오류가 발생했을 때는 1로 설정된다.



<카르노 맵>



<SOP>

AB \ CDE	000	001	011	010	110	111	101	100
00	0	1	0	1	0	1	0	1
01	1	0	1	0	1	0	1	0
11	0	1	0	1	0	1	0	1
10	1	0	1	0	1	0	1	0

<POS>

① SOP

1로 묶인 항들을 적어보면

A′B′C′D′E + A′B′C′DE′+A′B′CDE+A′B′CD′E′+A′BC′D′E′+A′BC′DE+A′BCDE′+

A′BCD′E+ABC′D′E+ABC′DE′+ABCDE+ABCD′E′+AB′C′D′E′+AB′C′DE+AB′CDE′+AB′CD′E

=

A′B′C′(D′E+DE′)+A′B′C(DE+D′E′)+A′BC′(D′E′+DE)+A′BC(DE′+D′E)+ABC′(D′E+DE′)+ABC(DE+D′E′)+AB′C′(D′E′+DE)+AB′C(DE′+D′E)

=A′B′C′(D⊕E) + A′B′C(D⊕E)′+A′BC′(D⊕E)′+A′BC(D⊕E) + ABC′(D⊕E)+ABC(D⊕E)′ + AB′C′(D⊕E)′ + AB′C(D⊕E)

=A′(D⊕E)(B′C′+BC) + A(BC′+B′C)(D⊕E) + A′(B′C+BC′)(D⊕E)′ + A(BC+B′C′)(D⊕E)′

=A′(D⊕E)(B⊕C)′ + A(B⊕C)(D⊕E) + A′(B⊕C)(D⊕E)′ + A(B⊕C)′(D⊕E)′

=A′(D⊕E) ⊕ (B⊕C) + A((B⊕C) ⊕ (D⊕E))′

=A⊕B⊕C⊕D⊕E 이다.

② POS

0으로 묶인 항들을 적어보면

보수 =

A′B′C′D′+A′B′C′DE+A′B′CDE′+A′B′CD′E+A′BC′D′E+A′BC′DE′+A′BCDE+A′BCD′E′+ABC′D′E′+ABC′DE+ABCDE′+ABCD′E+AB′C′D′E+AB′C′DE′+AB′CDE+AB′CD′E′

이므로 다시 보수를 취해주면,

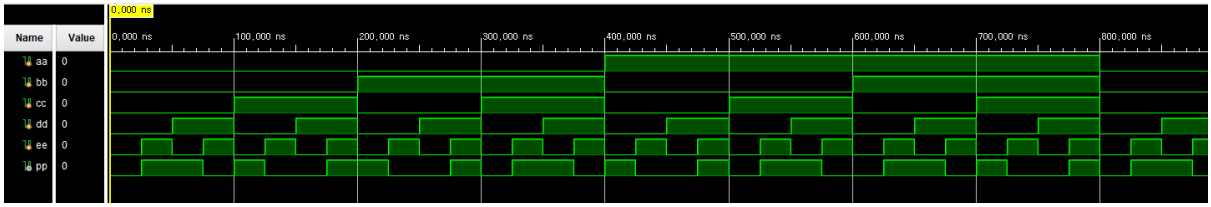
(A'B'C'D'E'+A'B'C'DE+A'B'CDE'+A'B'CD'E+A'BC'D'E+A'BC'DE'+A'BCDE+A'BCD'E'+ABC'D'E'+ABC'DE+ABCDE'+ABCD'E+AB'C'D'E+AB'C'DE'+AB'CDE+AB'CD'E')'

=

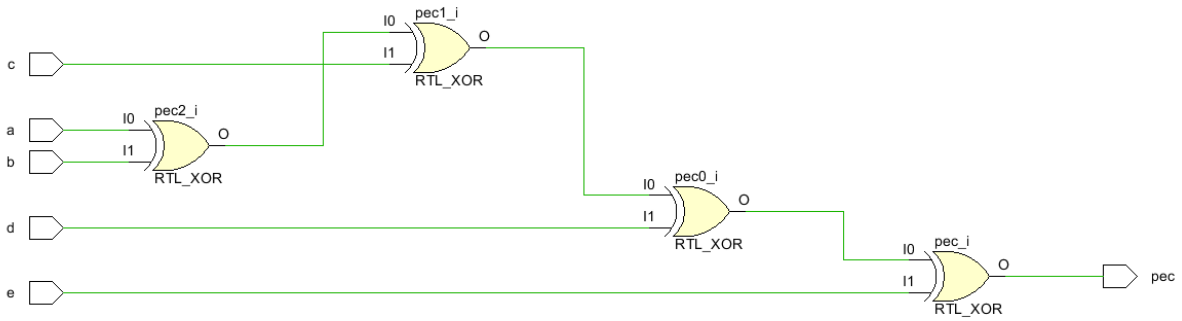
(A'B'C'D'E')'(A'B'C'DE)'(A'B'CDE')'(A'B'CD'E)'(A'BC'D'E)'(A'BC'DE')'(A'BCDE)'(A'BCD'E')(ABC'D'E')'(ABC'DE)'(ABCDE')'(ABCD'E)'(AB'C'D'E)'(AB'C'DE')'(AB'CDE)'(AB'CD'E')'

=

(A+B+C+D+E)(A+B+C+D'+E')(A+B+C'+D'+E)(A+B+C'+D+E')(A+B'C+D+E)(A+B'+C+D'+E)(A+B'+C'+D'+E')(A+B'+C'+D+E)(A'+B'+C+D+E)(A'+B'+C+D'+E')(A'+B'+C'+D'+E')(A'+B'+C'+D+E')(A'+B+C+D+E)(A'+B+C'+D'+E')(A'+B+C'+D+E) 이다.



<Simulation 결과>



<Schematic>

Even Parity Bit Checker의 진리표를 작성하고 진리표를 기반으로 카르노 맵을 작성하였다. 카르노 맵에서 1로 묶어서 표현한 SOP 방법과 0으로 묶어서 보수를 취하는 방식으로 표현한 POS 방법으로 식을 표현하였다. 이 때, SOP로 나타낸 식은 식의 정리를 통해서 input 끼리의 XOR 값인 A⊕B⊕C⊕D⊕E 이 된다는 것을 알 수 있었다.

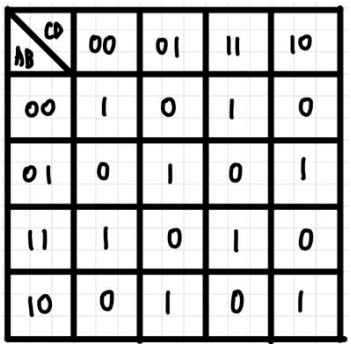
Simulation의 결과를 통해서 진리표와 동일한 결과가 출력된다는 것을 확인할 수 있었고, Schematic을 통해서도 확인해볼 수 있었다.

2. Odd Parity bit generator 및 checker의 simulation 결과 및 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)

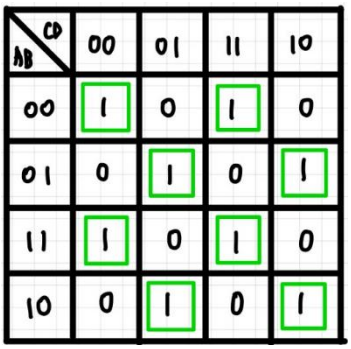
1) Odd Parity bit generator

Input A	Input B	Input C	Input D	Output E
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

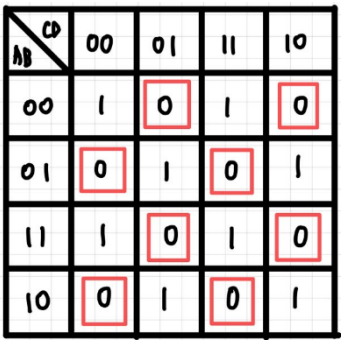
Odd Parity Bit인 경우 1의 개수가 홀수가 되도록 Parity Bit을 설정해주어야 한다. 즉, Input의 1의 개수가 홀수일 경우에는 Output이 0으로 설정되고, Input의 1의 개수가 짝수일 경우에는 Output이 1로 설정되어야 한다.



<카르노 맵>



<SOP>



<POS>

① SOP

1로 묶인 항들을 표현하면,

A′B′C′D′ + A′B′CD + A′BC′D+A′BCD′+ABC′D′+ABCD+AB′C′D+AB′CD′

= A′B′(C′D′+CD)+A′B(C′D+CD′)+AB(C′D′+CD)+AB′(C′D+CD′)

= A′B′(C⊕D)′ + A′B(C⊕D) + AB(C⊕D)′+AB′(C⊕D)

=(A′B′+AB)(C⊕D)′ + (A′B+AB′)(C⊕D)

=(A⊕B)′(C⊕D)′ + (A⊕B)(C⊕D)

=(A⊕B⊕C⊕D)′ 이다.

② POS

0으로 묶인 항들을 적어보면,

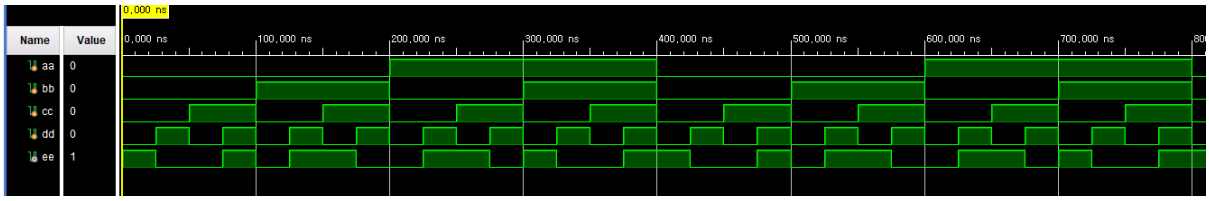
보수 = A′B′C′D + A′B′CD′+A′BC′D′+A′BCD+ABC′D+ABCD′+AB′C′D′+AB′CD

이므로 다시 보수를 취해주면,

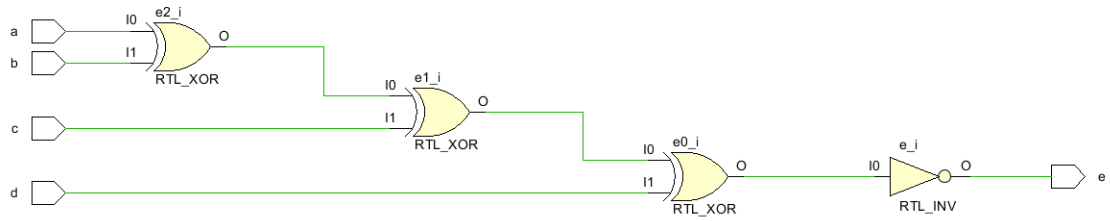
(A′B′C′D + A′B′CD′+A′BC′D′+A′BCD+ABC′D+ABCD′+AB′C′D′+AB′CD)′

=(A′B′C′D)′(A′B′CD′)′(A′BC′D′)′(A′BCD)′(ABC′D)(ABCD′)(AB′C′D′)(AB′CD)′

=(A+B+C+D)(A+B+C′+D)(A+B′+C+D)(A+B′+C′+D′)(A′+B′+C+D′)(A′+B′+C′+D)(A′+B+C+D)(A′+B+C′+D′) 이다.



<Simulation>



<Schematic>

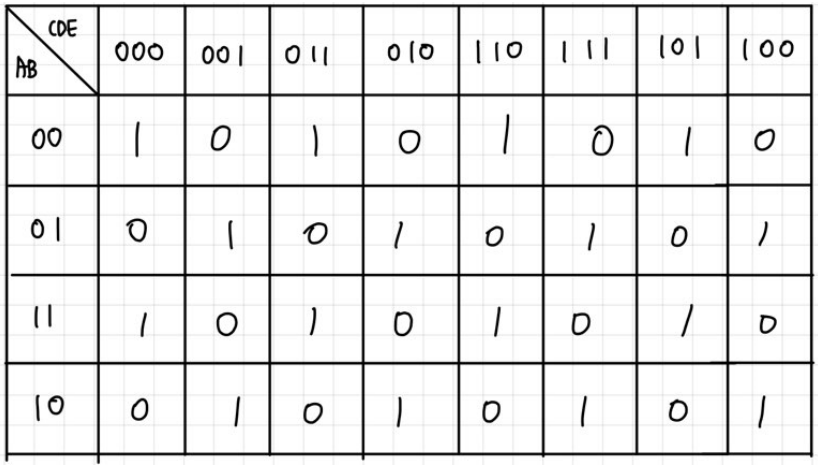
Odd Parity Bit Generator의 진리표를 작성하고 진리표를 기반으로 카르노 맵을 작성하였다. 카르노 맵에서 1로 묶어서 표현한 SOP 방법과 0으로 묶어서 보수를 취하는 방식으로 표현한 POS 방법으로 식을 표현하였다. 이 때, SOP로 나타낸 식은 식의 정리를 통해서 input 끼리의 XOR 값의 부정인 (A⊕B⊕C⊕D)'이 된다는 것을 알 수 있었다.

Simulation의 결과를 통해서 진리표와 동일한 결과가 출력된다는 것을 확인할 수 있었고, Schematic을 통해서도 확인해볼 수 있었다.

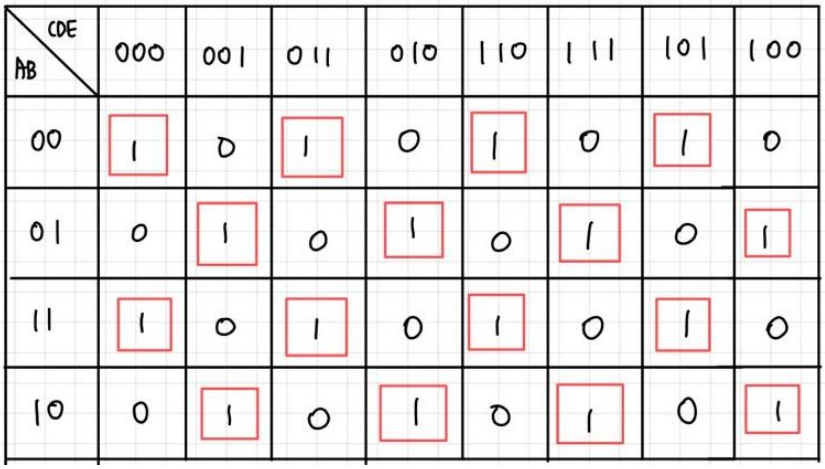
2) Odd Parity bit checker

Input A	Input B	Input C	Input D	Input E	Output 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	1	0	0	1
0	1	0	1	1	0
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	1	0	0	1
1	1	0	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	0	1	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	0	0	0

1	1	1	1	0	1
1	1	1	0	1	1
1	1	0	1	1	1
1	0	1	1	1	1
1	1	1	1	1	0



<카르노 맵>



<SOP>

AB \ CDE	000	001	011	010	110	111	101	100
00	1	0	1	0	1	0	1	0
01	0	1	0	1	0	1	0	1
11	1	0	1	0	1	0	1	0
10	0	1	0	1	0	1	0	1

<POS>

① SOP

1로 묶인 항들을 보면

A'B'C'D'E'+A'B'C'DE+A'B'CDE'+A'B'CD'E+A'BC'D'E+A'BC'DE'+A'BCDE+A'BCD'E'+ABC'D'E'+ABC'DE+ABCDE'+ABCD'E+AB'C'D'E+AB'C'DE'+AB'CDE+AB'CD'E'

=A'B'C'(D'E'+DE) + A'B'C(DE'+D'E)+A'BC'(D'E+DE')+A'BC(DE+D'E')+ABC'(D'E'+DE)+ABC(DE'+D'E) + AB'C'(D'E+DE')+AB'C(DE+D'E')

=A'B'C'(D⊕E)' + A'B'C(D⊕E) +

A'BC'(D⊕E) + A'BC(D⊕E)'+ABC'(D⊕E)'+ABC(D⊕E)+AB'C'(D⊕E)+AB'C(D⊕E)'

=A'(B'C'+BC)(D⊕E)' + A'(B'C+BC')(D⊕E) + A(BC'+B'C)(D⊕E)' + A(BC+B'C')(D⊕E)

=A'(B⊕C)'(D⊕E)' + A'(B⊕C)(D⊕E) + A(B⊕C)(D⊕E)' + A(B⊕C)'(D⊕E)

=A'(B⊕C⊕D⊕E)' + A(B⊕C⊕D⊕E)

= (A⊕B⊕C⊕D⊕E)' 이다.

② POS

0으로 항을 묶어서 보수를 구해보면,

보수 = A'B'C'D'E + A'B'CD'E +

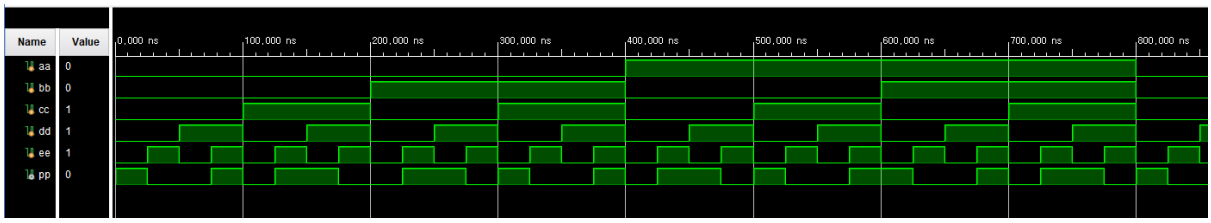
A'B'CDE+A'B'CD'E'+A'BC'D'E'+A'BC'DE+A'BCDE'+A'BCD'E+ABC'D'E+ABC'DE'+ABCDE+ABCD'E'+AB'C'D'E'+AB'C'DE+AB'CDE'+AB'CD'E 이므로, 다시 보수를 취해주면

(A′B′CDE+A′B′CD′E′+A′BC′D′E′+A′BC′DE+A′BCDE′+A′BCD′E+ABC′D′E+ABC′DE′+ABCDE+ABCD′E′+AB′C′D′E′+AB′C′DE+AB′CDE′+AB′CD′E)′

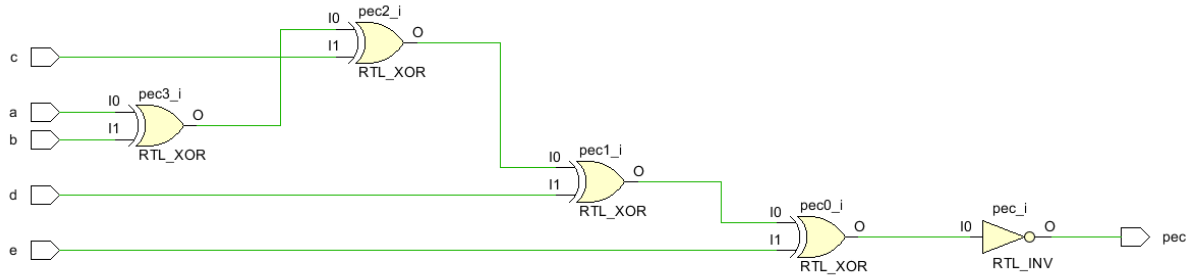
=(A′B′CDE)′(A′B′CD′E′)′(A′BC′D′E′)′(A′BC′DE)′(A′BCDE)′(A′BCD′E)′(ABC′D′E)′(ABC′DE′)′(ABCDE)′(ABCD′E′)′

(AB′C′D′E′)′(AB′C′DE)′(AB′CDE′)′(AB′CD′E)′

=(A+B+C′+D′+E′)(A+B+C′+D+E)(A+B′+C+D+E)(A+B′+C+D′+E′)(A+B′+C′+D′+E′)(A+B′+C′+D+E′)(A′+B′+C+D+E′)(A′+B′+C+D′+E)(A′+B′+C′+D′+E′)(A′+B′+C′+D+E)(A′+B+C+D+E)(A′+B+C+D′+E′)(A′+B+C′+D′+E)(A′+B+C′+D+E) 이다.



<Simulation>



<Schematic>

Odd Parity Bit Checker의 진리표를 작성하고 진리표를 기반으로 카르노 맵을 작성하였다. 카르노 맵에서 1로 묶어서 표현한 SOP 방법과 0으로 묶어서 보수를 취하는 방식으로 표현한 POS 방법으로 식을 표현하였다. 이 때, SOP로 나타낸 식은 식의 정리를 통해서 input 끼리의 XOR 값의 부정인 (A⊕B⊕C⊕D⊕E)′ 이 된다는 것을 알 수 있었다.

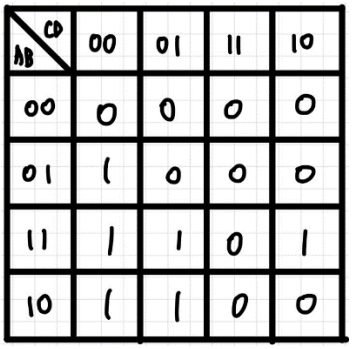
Simulation의 결과를 통해서 진리표와 동일한 결과가 출력된다는 것을 확인할 수 있었고, Schematic을 통해서도 확인해볼 수 있었다.

3. 2-bit binary comparator simulation 결과 및 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)

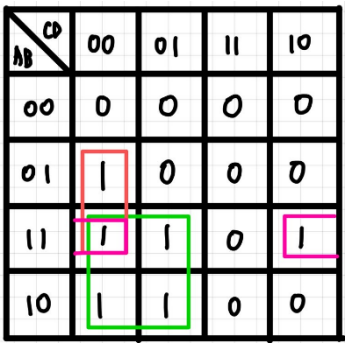
Input A	Input B	Input C	Input D	Output F1	Output F2	Output F3
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

2-bit binary comparator에서 Input A와 Input B는 첫번째 숫자의 각 자리를 의미하며, Input C와 Input D는 두번째 숫자의 각 자리를 의미한다. 따라서 Output F1은 첫번째 숫자가 더 클 때 1로 설정되며, Output F2는 두 숫자가 같을 때 1로 설정되고, Output F3는 두번째 숫자가 더 클 때 1로 설정된다.

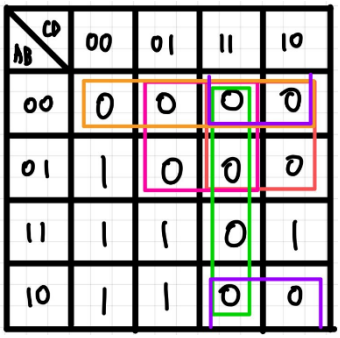
1) Output F1



<카르노 맵>



<SOP>



<POS>

① SOP (NAND형태로 표현)

BC'D'+ABD'+AC' = ((BC'D')'(ABD')'(AC')')'

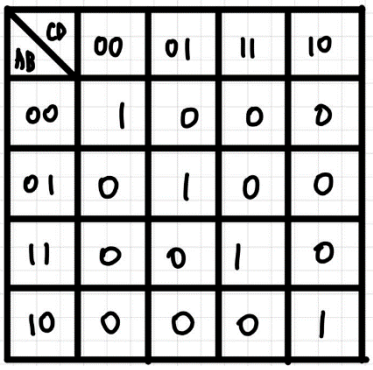
②POS

0으로 묶어서 보수를 구해보면,

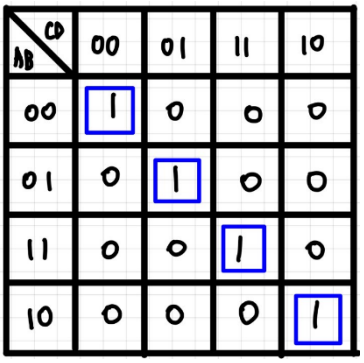
보수 = A'B'+A'D+A'C+CD+B'C 이므로 다시 보수를 취해주면,

(A'B'+A'D+A'C+CD+B'C)' = (A'B')'(A'D)'(A'C)'(CD)'(B'C')' = (A+B)(A+D')(A+C')(C'+D')(B+C) 이다.

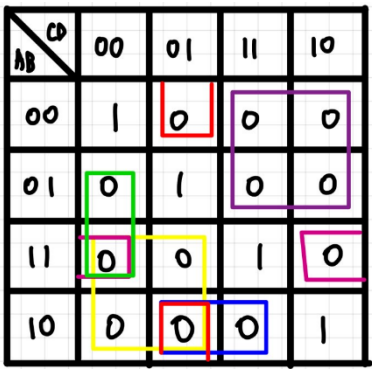
2) Output F2



<카르노 맵>



<SOP>



<POS>

① SOP (NAND 형태로 표현)

A'B'C'D' + A'BC'D + ABCD + AB'CD' = ((A'B'C'D')'(A'BC'D)'(ABCD)'(AB'CD')')'

②POS

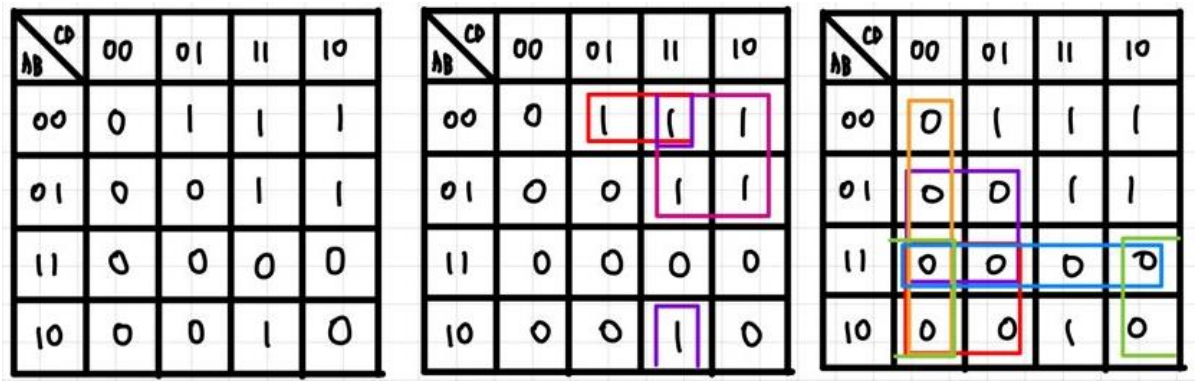
0으로 묶어서 보수를 구해보면,

보수 = BC'D'+ABD'+AC'+AB'D+B'C'D+A'C 이므로 다시 보수를 취해주면,

(BC'D'+ABD'+AC'+AB'D+B'C'D+A'C)' = (BC'D')'(ABD')'(AC')'(AB'D)'(B'C'D)'(A'C)'

= (B'+C+D)(A'+B'+D)(A'+C)(A'+B+D')(B+C+D')(A+C') 이다.

3) Output F3



<카르노 맵> <SOP> <POS>

① SOP (NAND 형태로 표현)

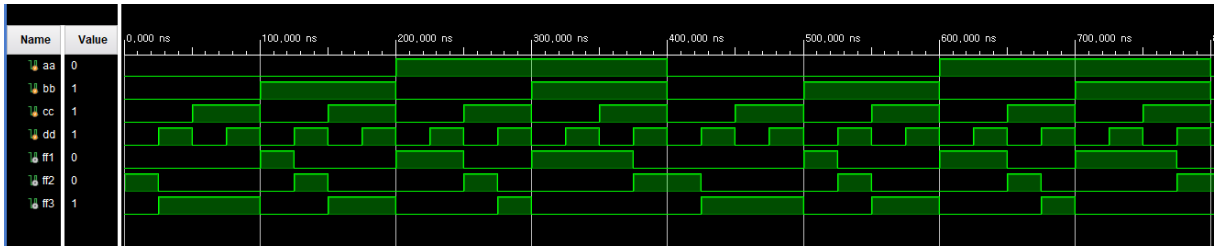
A'C+A'B'D+B'CD = ((A'C)'(A'B'D)'(B'CD)')'

②POS

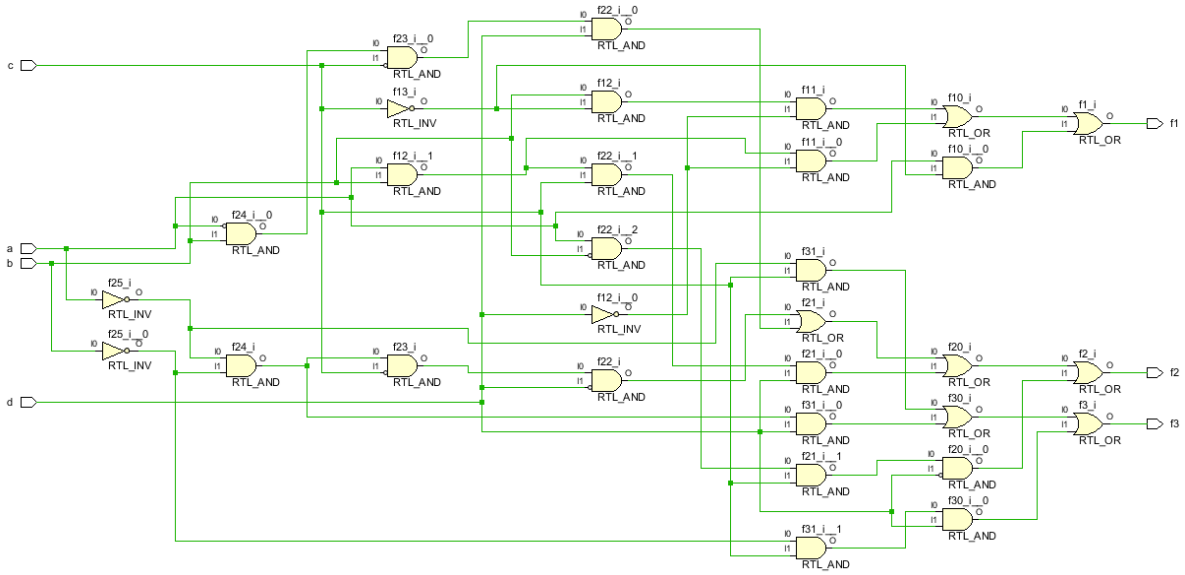
0으로 묶어서 보수를 구해보면,

보수 = C'D'+BC'+AC'+AD'+AB 이므로 다시 보수를 취해주면,

(C'D'+BC'+AC'+AD'+AB)' = (C'D')'(BC')'(AC')'(AD')'(AB)' = (C+D)(B'+C)(A'+C)(A'+D)(A'+B') 이다.



<Simulation>



<Schematic>

2-bit binary comparator의 output인 f1, f2, f3의 진리표를 각각 작성하고 진리표를 기반으로 카르노 맵을 작성하였다. 카르노 맵에서 1로 묶어서 표현한 SOP 방법과 0으로 묶어서 보수를 취하는 방식으로 표현한 POS 방법으로 식을 표현하였다. 표현한 식을 이용해서 Verilog 코딩을 진행하였다. Simulation의 결과를 통해서 진리표와 동일한 결과가 출력된다는 것을 확인할 수 있었고, Schematic을 통해서도 확인해볼 수 있었다.

4. 결과 검토 및 논의 사항

Even Parity Bit Generator의 경우 1의 개수가 짝수 개가 되어 전송될 수 있도록 Parity Bit을 설정한다. 즉, Parity Bit을 포함하여 1의 개수가 짝수가 되어야 한다. Even Parity Bit Checker은 Even Parity Bit을 검사해서 Parity Bit이 제대로 되어있는지 확인한다. 즉 입력된 값에서 1의 개수가 홀수일 경우 오류가 발생한 것이기 때문에 출력값을 1로 설정하고 올바르게 설정되었을 경우에 0으로 설정하게 된다. 이를 진리표와 카르노 맵을 통해서 수식을 결정해보았고 시뮬레이션을 통해서 확인해보았다. Odd Parity Bit Generator와 Odd Parity Bit Checker는 짝수로 설정되도록 하며 전체적인 흐름은 동일하다.

2-bit binary comparator의 경우 두 숫자를 비교하며, 비교되는 두 수가 A,B일 때, A>B이면 F1이 1로 설정되고, A=B일 경우에 F2가 1로 설정되며 A<B일 때 F3가 1로 설정된다. 이를 카르노 맵을 통해서 살펴보고 식을 간단히 하여 Verilog 코딩을 해보았고 시뮬레이션을 통해서 결과를 확인해보았다.

5. 추가 이론 조사 및 작성

2-bit binary comparator의 경우, 이와 비슷한 방식으로 bit 수를 확대해서 N-bit comparator를 생각해볼 수 있다. 각각의 자릿수를 비교하여 전체 숫자의 크기를 비교해볼 수 있다.

수학에서 Parity란 짝수, 홀수를 의미하며, 이진수일 때는 최하위 비트에 의해서 결정이 되게 된다. 통신에 있어서, Parity는 주어진 비트 내에서 비트 수에 대한 정보를 담고 있으며 모든 비트의 값에 의해서 결정된다. 비트의 XOR합계를 통해서 계산할 수 있다.

패리티 비트는 단순히 오류 발생 여부만 알 수 있는 반면에, 이를 응용해서 오류를 정정할 수 있는 코드가 존재하며, 해밍코드가 그 예시이다. 해밍코드의 경우 하나의 비트에 오류가 발생했을 대 정정할 수 있다. 반면에 2개 이상의 비트에 오류가 발생할 경우 정정할 수 없다는 단점이 있다.