6주차 결과보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

▪ Adder(가산기)와 Subtractor(감산기)의 개념을 이해한다.

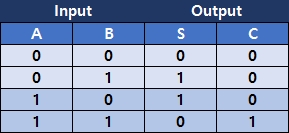
▪ Code converter(부호 변환기)의 개념을 이해한다.

▪ Verilog를 사용하여 다양한 Adder와 Subtractor를 구현한다.

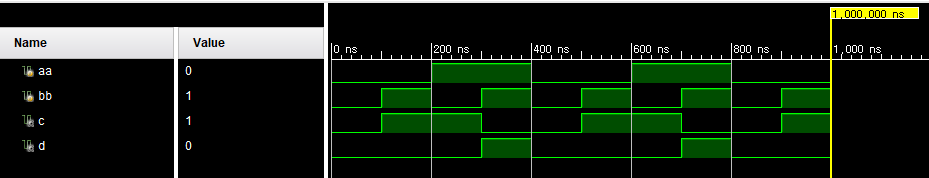
▪ Verilog를 사용하여 다양한 Code converter를 구현한다.

▪ FPGA 통해서 Verilog로 구현된 회로의 동작을 확인한다.

**2.**



[Half Adder의 진리표]

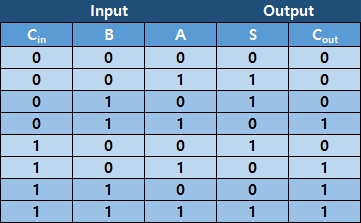


[Half Adder의 Simulation 결과] – (aa, bb, c, d가 각각 A, B, S, C에 대응됨)

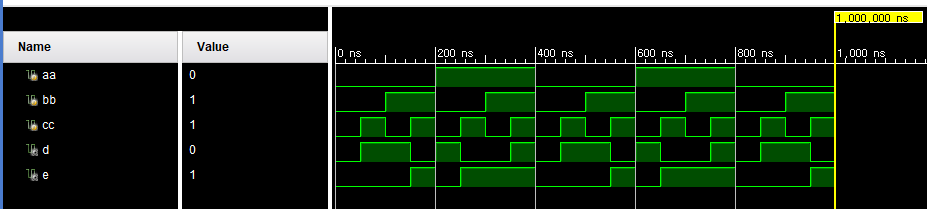
반가산기(Half Adder)는 입력되는 1비트의 두 이진수 A, B를 더하여 합 S(Sum)와 올림수 C(Carry)를 구하는 회로이다.

위 진리표에서 확인할 수 있듯이, S의 값은 A와 B가 같은 값을 갖는 경우 0을, 다른 값을 갖는 경우 1을 가지는 것을 확인할 수 있다. 이는 XOR과 동치이므로 로 나타낼 수 있다.

C의 값은 A와 B가 모두 1일때만 1을 갖는데, 이는 AND와 동치이므로 로 나타낼 수 있다.



[Full Adder의 진리표]



[Full Adder의 Simulation 결과] – (aa, bb, cc, d, e가 각각 C(in), B, A, S, C(out)에 대응됨)

전가산기(Full Adder)는 입력되는 1비트의 두 이진수 A, B와 이전 자리의 올림수 을

더하여 합 S와 올림수 을 구하는 회로이다.

위 진리표에서 확인할 수 있듯이, S의 값은 입력값 (A, B, )들 중 1의 개수가 홀수개

일 때 1을, 짝수개 일 때 0을 갖는데, 이는 3-input XOR 게이트와 동치이므로

로 나타낼 수 있다.

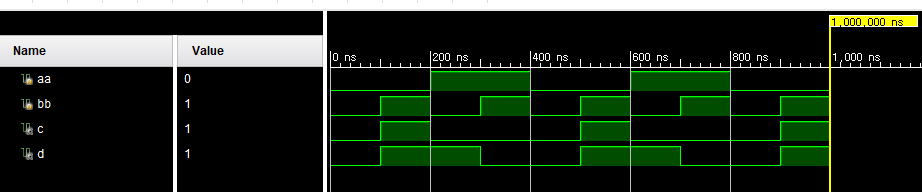
의 값은 A와 B가 모두 1이면 값과 관계없이 1이 되고(1), 이 1이고, A와 B 중 하나가 1인 경우 1이 된다(2). (1)의 경우 수식으로 나타내면 이고, (2)의 경우 수식으로 나타내면 인데, 은 (1) 또는 (2)일 때 1이 되므로 OR을 사용해

로 나타낼 수 있다.

**3.**



[Half Subtractor의 진리표]



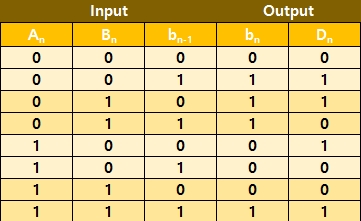
[Half Subtractor의 Simulation 결과] – (aa, bb, c, d가 각각 A, B, b, D에 대응됨)

반감산기(Half Subtractor)는 입력되는 1비트의 두 이진수 A, B를 뺄셈하여

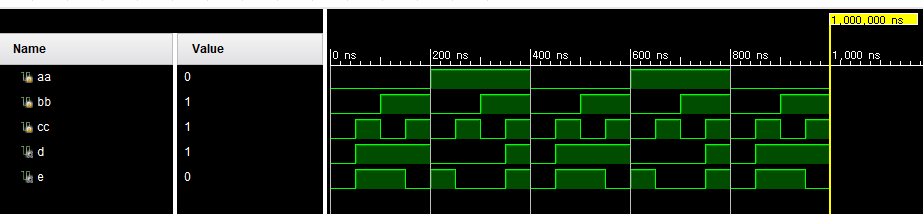
차 D(Difference)와 빌림수 b(borrow)를 구하는 회로이다.

위 진리표에서 확인할 수 있듯이, D의 값은 A와 B가 같은 값을 갖는 경우 0을, 다른 값을 갖는 경우 1을 가지는 것을 확인할 수 있다. 이는 XOR과 동치이므로 로 나타낼 수 있다.

b의 값은 A가 0, B가 1일때만 1을 갖는데, 이는 가 1, B가 1일때만 1을 갖는다고 볼 수 있다. 이는 AND와 동치이므로 로 나타낼 수 있다.



[Full Subtractor의 진리표]



[Full Subtractor의 Simulation 결과] – (aa, bb, cc, d, e가 각각 A, B, b(n-1), b(n), D에 대응됨)

전감산기(Full Subtractor)는 입력되는 1비트의 두 이진수 A, B와 이전 자리의 빌림수

를 포함하여 차 D와 빌림수 을 구하는 회로이다.

위 진리표에서 확인할 수 있듯이, D의 값은 입력값 (A, B, )들 중 1의 개수가 홀수개

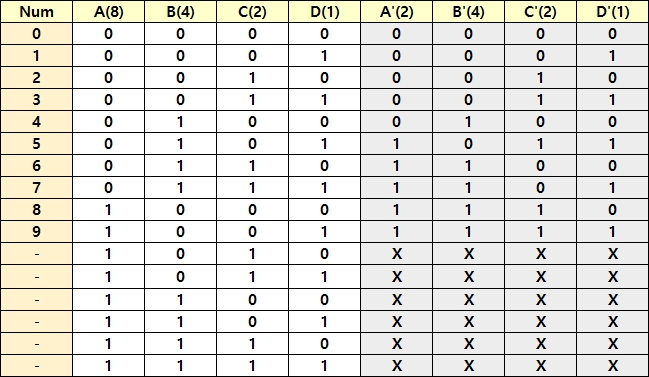
일 때 1을, 짝수개 일 때 0을 갖는데, 이는 3-input XOR 게이트와 동치이므로

로 나타낼 수 있다.

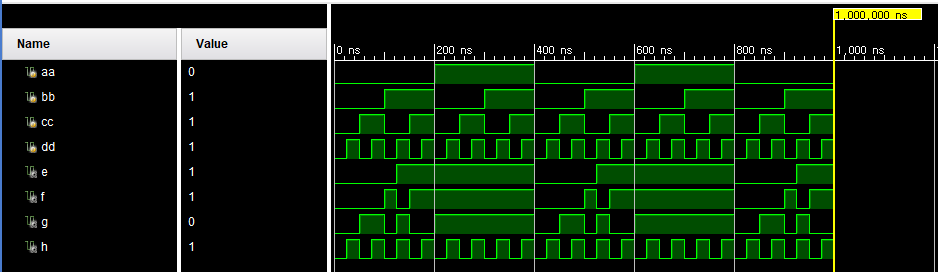
의 값은 B가 A보다 크면 값과 관계없이 1이 되고(1), 이 1이고, A와 B가 같은 값을 갖는 경우 1이 된다(2). (1)의 경우 수식으로 나타내면 이고 (반감산기의 b와 같은 경우), (2)의 경우 수식으로 나타내면 인데 (A와 B가 같을 때만 1이 되는 경우는 XNOR과 동치), 은 (1) 또는 (2)일 때 1이 되므로 OR을 사용해

로 나타낼 수 있다.

**4.**



[8421-2421 Code converter의 진리표]



[8421-2421 Code converter 의 Simulation 결과] – (aa, bb, cc, dd, e, f, g, h가 각각 A, B, C, D, A’, B’, C’, D’에 대응됨)

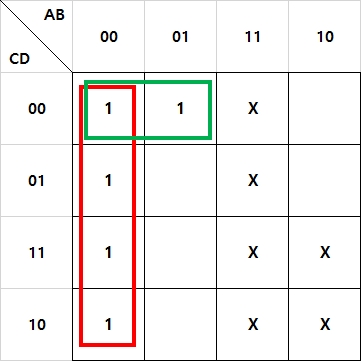
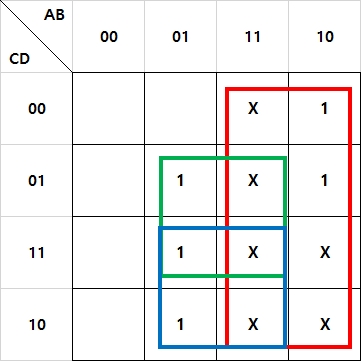
8421과 2421 Code는 0부터 9까지의 수만을 표시할 수 있기 때문에 8421과 2421 Code에서 4비트로 나타낼 수 있는 나머지의 이진수들은 사용되지 않는다. 이에 따라 8421에서 2421 Code로 변환할 때 0~9가 아닌 경우는 변환 후 X(don’t care condition)으로 표기하였다.

위 진리표에서 8421 Code를 표현하는 4비트의 이진수를 MSB부터 순서대로 A, B, C, D로 두고,

변환된 2421 Code를 표현하는 4비트의 이진수를 MSB부터 순서대로 A’, B’, C’, D’라 하면,

8421-2421 Code converter는 4-input, 4-output 회로라고 할 수 있다. 출력값 A’, B’, C’, D’의

입력값 A, B, C, D에 대한 논리 함수를 카르노맵을 사용해 SOP form과 POS form로 각각 구하면 다음과 같다.



[Output A’의 논리 함수 카르노맵 (오른쪽은 A’의 부정)]

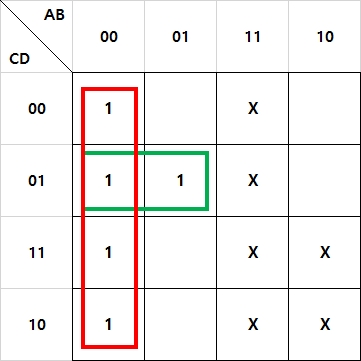
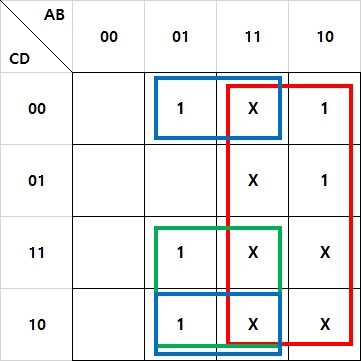
왼쪽의 카르노 맵에서, SOP를 구하기 위해 진리표에서 A’이 1이 되는 부분과 don’t care 부분을 카르노맵에 표기하였다. 각각의 박스는 같은 색깔끼리 하나의 주항(prime implicant)으로 묶인다.

따라서 A’의 SOP : 이다.

오른쪽의 카르노 맵에서, POS를 구하기 위해 진리표에서 A’의 부정을 카르노맵에 나타내었다.

A’의 부정의 SOP : 이고, 에서 A’의 POS :

이다.



[Output B’의 논리 함수 카르노맵 (오른쪽은 B’의 부정)]

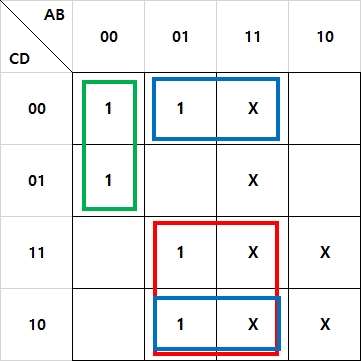
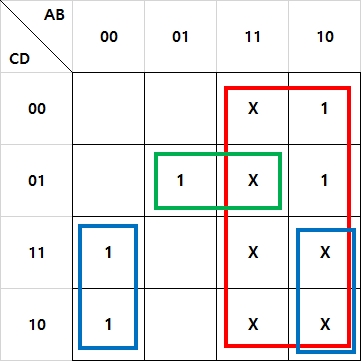
왼쪽의 카르노 맵에서, SOP를 구하기 위해 진리표에서 B’이 1이 되는 부분과 don’t care 부분을 카르노맵에 표기하였다. 각각의 박스는 같은 색깔끼리 하나의 주항(prime implicant)으로 묶인다.

따라서 B’의 SOP : 이다.

오른쪽의 카르노 맵에서, POS를 구하기 위해 진리표에서 B’의 부정을 카르노맵에 나타내었다.

B’의 부정의 SOP : 이고, 에서 B’의 POS :

이다.



[Output C’의 논리 함수 카르노맵 (오른쪽은 C’의 부정)]

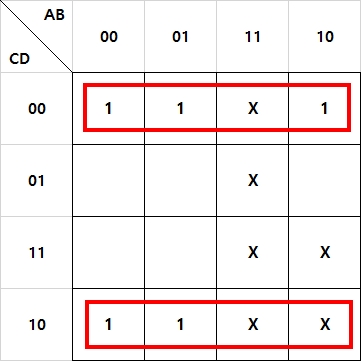
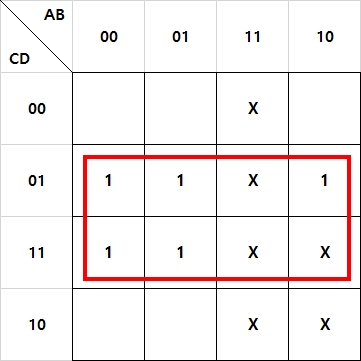
왼쪽의 카르노 맵에서, SOP를 구하기 위해 진리표에서 C’이 1이 되는 부분과 don’t care 부분을 카르노맵에 표기하였다. 각각의 박스는 같은 색깔끼리 하나의 주항(prime implicant)으로 묶인다.

따라서 C’의 SOP : 이다.

오른쪽의 카르노 맵에서, POS를 구하기 위해 진리표에서 C’의 부정을 카르노맵에 나타내었다.

C’의 부정의 SOP : 이고, 에서 C’의 POS :

이다.



[Output D’의 논리 함수 카르노맵 (오른쪽은 D’의 부정)]

왼쪽의 카르노 맵에서, SOP를 구하기 위해 진리표에서 D’이 1이 되는 부분과 don’t care 부분을 카르노맵에 표기하였다. 각각의 박스는 같은 색깔끼리 하나의 주항(prime implicant)으로 묶인다.

따라서 D’의 SOP : 이다.

오른쪽의 카르노 맵에서, POS를 구하기 위해 진리표에서 D’의 부정을 카르노맵에 나타내었다.

D’의 부정의 SOP : 이고, 에서 D’의 POS : 이다.

**5.**

2.~ 4.에서 각각의 시뮬레이션 결과와 진리표를 비교했을 때 차이가 없으므로 의도한 대로

코드가 작성되었음을 확인 할 수 있다.

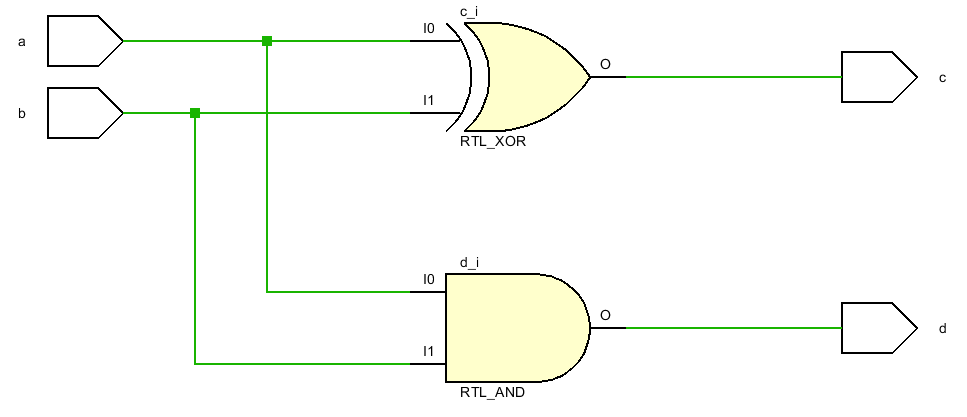
이 실험을 통해 전가산기, 반가산기, 전감산기, 반감산기의 구조와 작동방식(출력 값들의 진리표와 논리 함수 등)에 대해 이해할 수 있었다.

또한, 8421-2421 Code Converter의 논리함수를 직접 카르노맵으로 구함으로써 복잡한

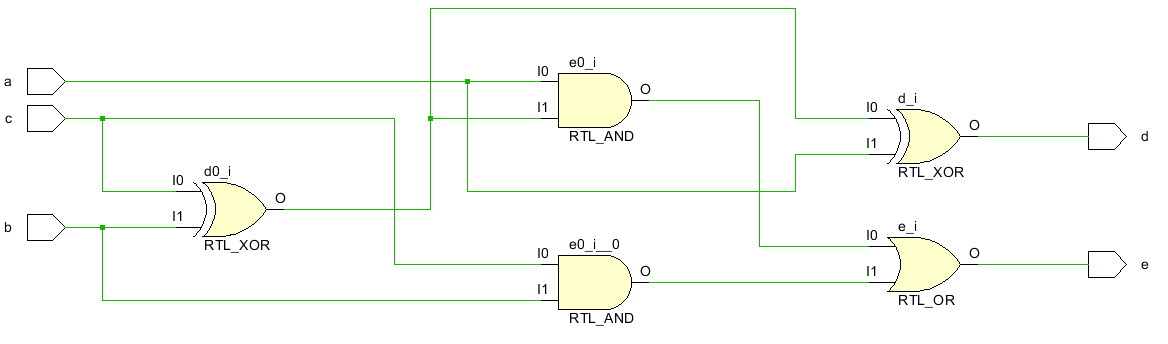
진리표를 갖는 Boolean 함수를 카르노맵을 사용하여 SOP와 POS 형태의 관계식을

구할 수 있음을 알게되었다.

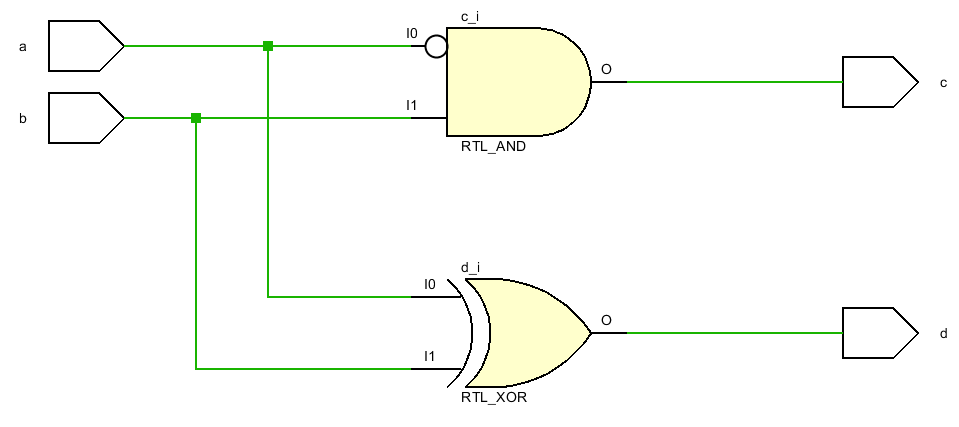
아래의 사진들은 2. ~ 4. 각각의 Schematic이다.



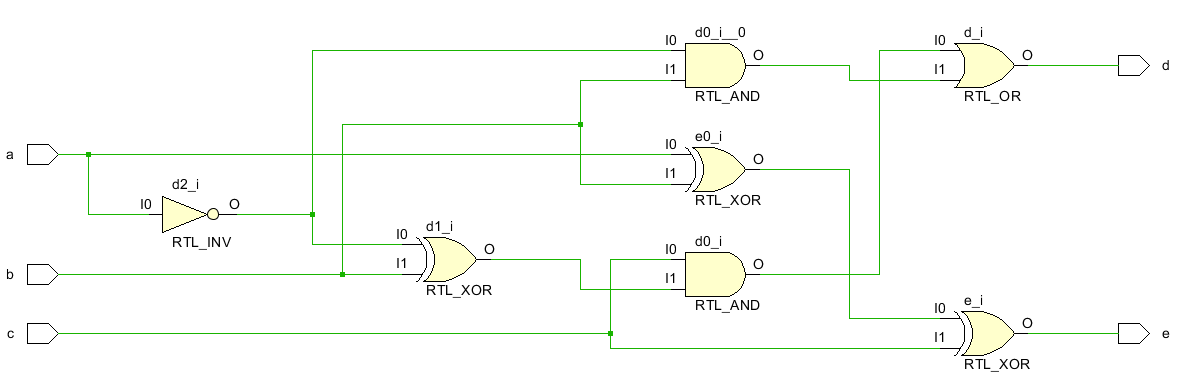
[Half Adder의 Schematic]



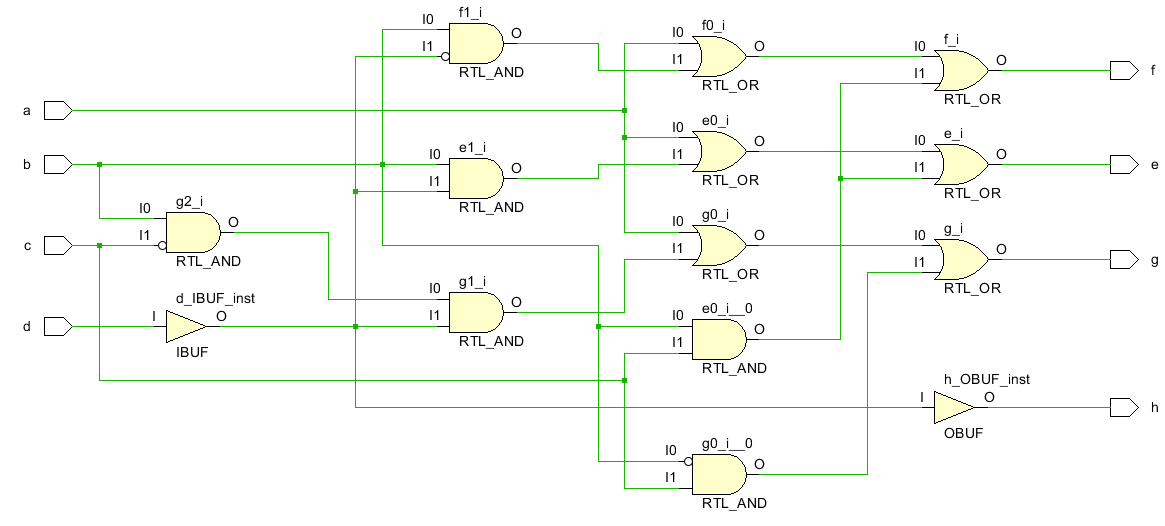
[Full Adder의 Schematic]



[Half Subtractor의 Schematic]



[Full Subtractor의 Schematic]



[8421-2421 Code Converter의 Schematic]

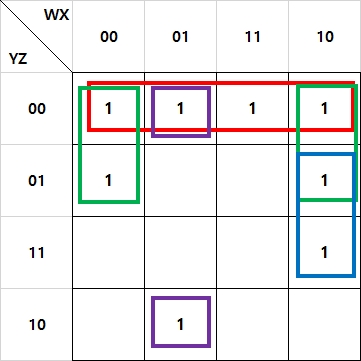
**6.**

- SOP form에서 POS form으로 변환하는 방법

SOP form에서 POS form으로 변환하는 방법에는 여러가지가 있다. 그 중 한 가지는 POS와 SOP가 서로 보수 관계임을 이용해 이중 부정의 형태로 POS form을 구하는 방법이다.

예를 들어, 를 POS 형태로 바꾼다고 한다면,

f’를 카르노맵으로 나타내면 아래와 같다.



[f’의 카르노맵 (같은 색깔끼리 주항을 이룸)]

카르노맵에서 구한 f’의 SOP는 이다.

여기서 (f’)’ = f이므로 이 특성을 사용하여 POS form을 구하면,

가 된다.