9주차 결과보고서

전공: 심리학과 학년: 3학년 학번: 20190345 이름: 김동현

**1.**

2 to 4 decoder는 2개의 입력에 대해 4개의 출력 중 하나의 신호만 출력한다. And gate로 구성한 decoder는 active high decoder로 선택된 출력은 1로, 선택되지 않은 출력은 0으로 출력된다. 아래 진리표는 2 to 4 and decoder의 진리표이다.

테이블이(가) 표시된 사진

자동 생성된 설명

진리표를 바탕으로 카르노맵을 구성한다.

D0에 대한 카르노맵을 표현하면 다음과 같다.

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

D0=a’b’

D1에 대한 카르노맵을 표현하면 다음과 같다.

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

D1=a’b

D2에 대한 카르노맵을 표현하면 다음과 같다.

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

D2=ab’

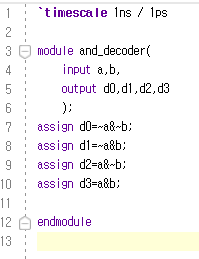
D3에 대한 카르노맵을 표현하면 다음과 같다.

텍스트, 시계이(가) 표시된 사진

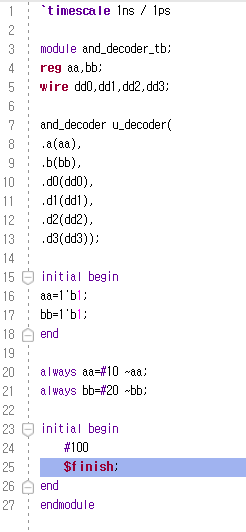
자동 생성된 설명

D3=ab

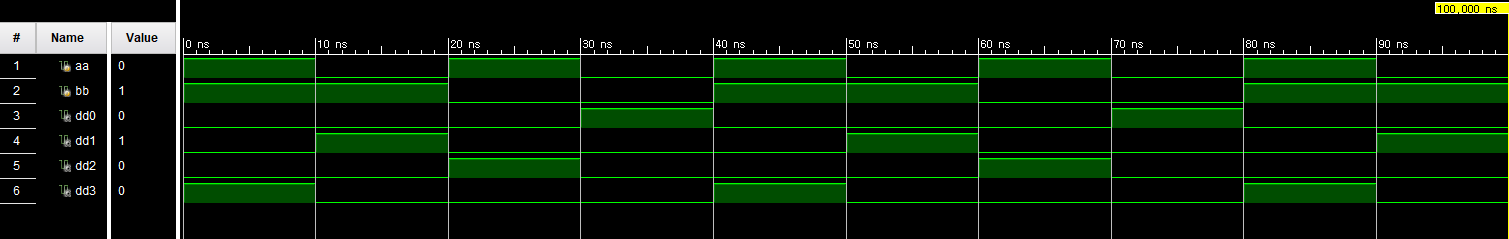
카르노맵을 통해 간소화한 논리식을 베릴로그를 통해 구현하였다. 디자인 소스 파일에서 2개의 입력(a,b)와 4개의 출력(d0,d1,d2,d3)를 선언하고, 카르노맵을 통해 구한 논리식을 각각 할당하였다.



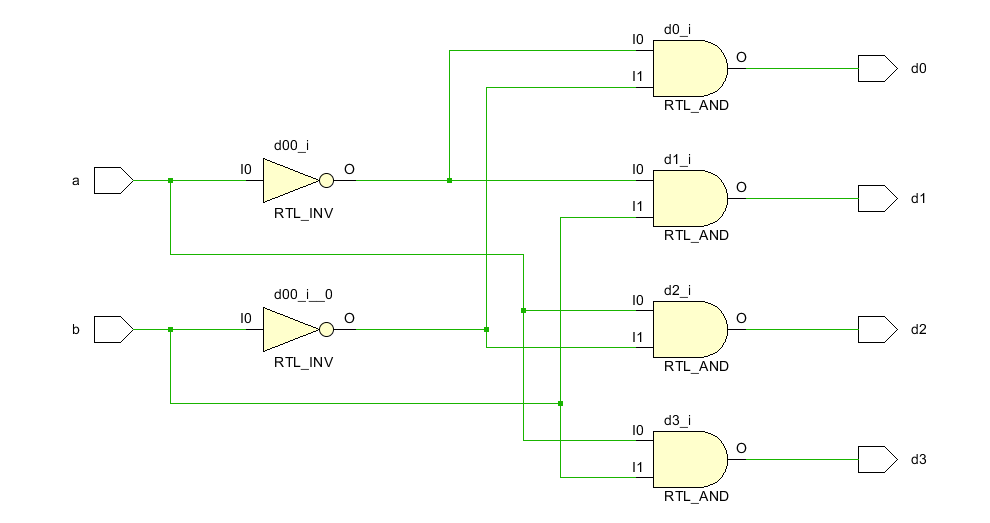
시뮬레이션 소스 파일을 다음과 같이 작성하였다.



시뮬레이션 결과는 아래의 그림과 같으며, 입력값 aa,bb의 값에 따라 하나의 결과값이 선택되어 출력(1)되는 것을 확인할 수 있다.



2 to 4 and decoder의 diagram을 Schematic으로 나타내었다.



Nand gate로 구성한 decoder는 active low decoder로 선택된 출력은 0로, 선택되지 않은 출력은 1으로 출력된다. 아래 진리표는 2 to 4 and decoder의 진리표이다.

테이블이(가) 표시된 사진

자동 생성된 설명

진리표를 바탕으로 카르노맵을 구성한다.

D0에 대한 카르노맵을 표현하면 다음과 같다.

텍스트, 시계, 탑재, 광장이(가) 표시된 사진

자동 생성된 설명

D0=~(~a&~b)

D1에 대한 카르노맵을 표현하면 다음과 같다.

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

D1=~(~a&b)

D2에 대한 카르노맵을 표현하면 다음과 같다.

텍스트, 시계, 시간, 탑재이(가) 표시된 사진

자동 생성된 설명

D2=~(a&~b)

D3에 대한 카르노맵을 표현하면 다음과 같다.

텍스트, 시계, 탑재이(가) 표시된 사진

자동 생성된 설명

D3=~(a&b)

카르노맵을 통해 간소화한 논리식을 베릴로그를 통해 구현하였다. 디자인 소스 파일에서 2개의 입력(a,b)와 4개의 출력(d0,d1,d2,d3)를 선언하고, 카르노맵을 통해 구한 논리식을 각각 할당하였다.

테이블이(가) 표시된 사진

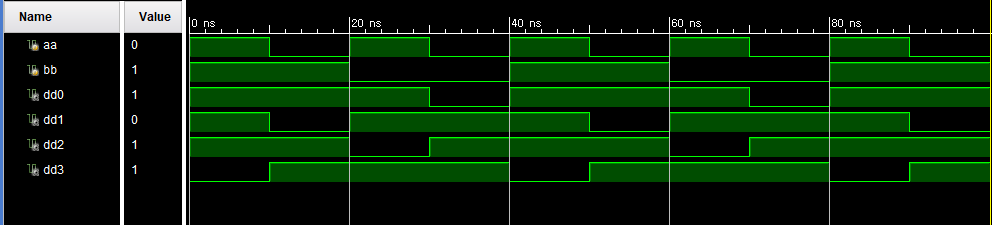
자동 생성된 설명

시뮬레이션 소스 파일을 다음과 같이 작성하였다.

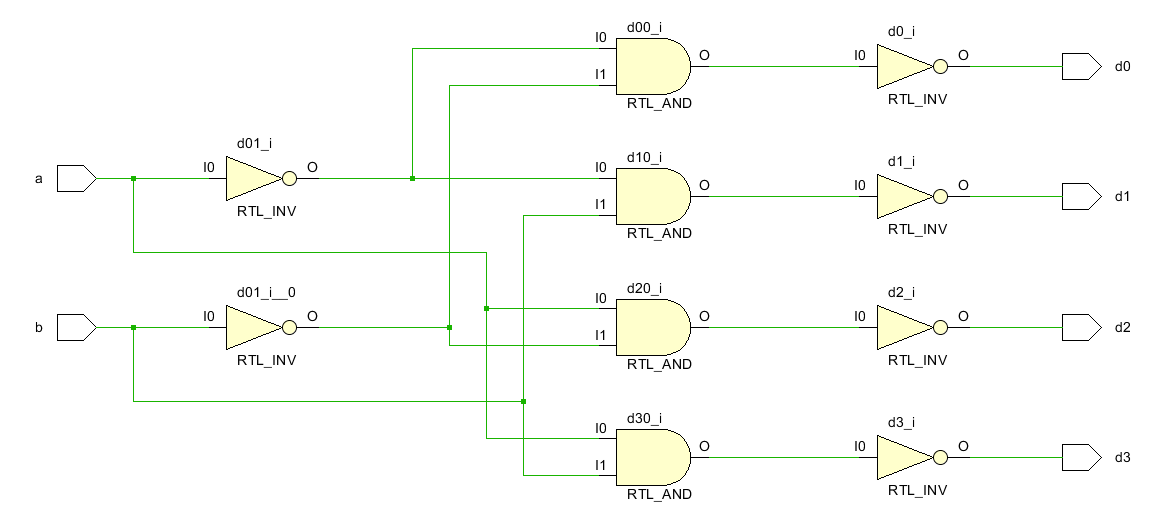
테이블이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 결과는 아래의 그림과 같으며, 입력값 aa,bb의 값에 따라 하나의 결과값이 선택되어 출력(0)되는 것을 확인할 수 있다.



2 to 4 nand decoder의 diagram을 Schematic으로 나타내었다.



2.

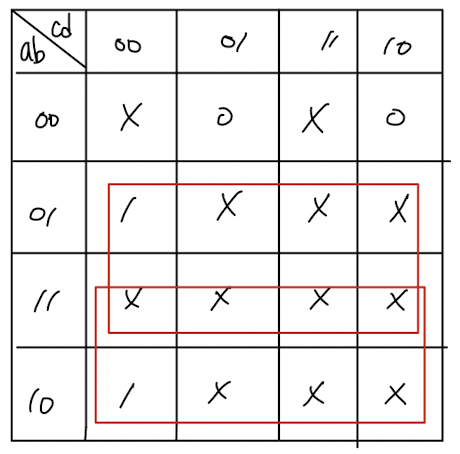
4 to 2 encoder는 decoder의 반대 기능을 수행하는 회로로서 4개의 입력에 대해 2개의 출력을 통해 신호를 출력한다. Or gate로 구성한 encoder는 디코더에서의 출력값이 입력값으로, 디코더에서의 입력값이 출력값으로 출력된다. 아래 진리표는 4 to 2 encoder의 진리표이다.

테이블이(가) 표시된 사진

자동 생성된 설명

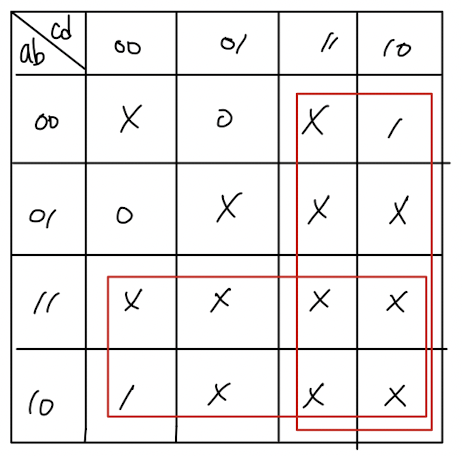
진리표를 바탕으로 카르노맵을 구성한다.

E0에 대한 카르노맵을 표현하면 다음과 같다.



E0=a|b

E1에 대한 카르노맵을 표현하면 다음과 같다.



E1=a|c

카르노맵을 통해 간소화한 논리식을 베릴로그를 통해 구현하였다. 디자인 소스 파일에서 4개의 입력(a,b,c,d)와 2개의 출력(E0,e1)를 선언하고, 카르노맵을 통해 구한 논리식을 각각 할당하였다.

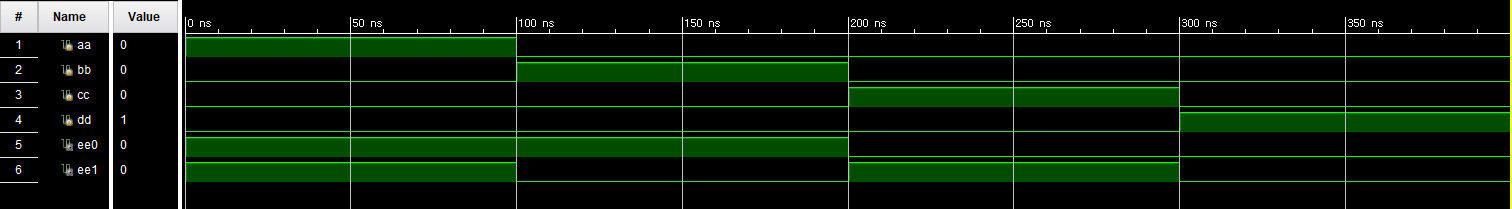
테이블이(가) 표시된 사진

자동 생성된 설명

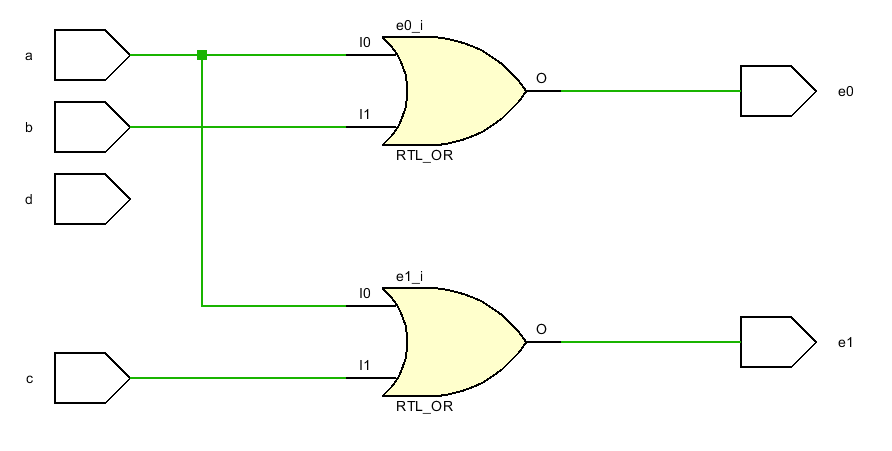
시뮬레이션 소스 파일을 다음과 같이 작성하였다.

테이블이(가) 표시된 사진

자동 생성된 설명



4 to 2 Encoder의 diagram을 Schematic으로 나타내었다.



3.

4개의 입력 중 4가지 형태(1000,0100,0010,0001)을 제외한 나머지 형태는 다음과 같이 나눌 수 있다. 우선 1이 하나도 입력되지 않는 경우(0000)은 모든 출력이 0으로 출력된다. 이러한 경우 0001과 같은 출력값이므로, 정상적인 입력(0001)이 출력된 것인지, 비정상적인 입력(0000)이 출력된 것인지 알 수 없는 모호성이 발생한다. 다른 경우는 1이 두개 이상 입력되는 경우(0011, 1100 등등)이다. 이러한 경우 각각의 입력에 대한 출력값이 나타나는 것이 아니다. 예를 들어 0011과 0110이 입력으로 주어진다면 출력으로 11이 출력될 것이다. 하지만 이는 1000에 대한 출력값이다. 이처럼 2개 이상의 1이 입력으로 들어온다면 올바른 출력이 출력되지 않는다.

4.

위와 같은 오류를 예방하기 위해서 priority encoder를 활용할 수 있다. 이는 입력 비트의 우선순위를 설정하여 오류의 발생을 막는다.

테이블이(가) 표시된 사진

자동 생성된 설명

1이 나오지 않은 상황에서는 오류 발생 여부를 나타내는 v값을 0으로 출력해 오류 발생 여부를 나타낸다. 그렇지 않은 경우에는 a부터 d까지 우선순위를 부여해 해당 입력이 1이라면 그 뒤 입력과 관계없이(don’t care) 해당 입력에 대한 출력값을 출력한다. 위의 진리표를 카르노 맵으로 나타내면 다음과 같다.

텍스트, 낱말맞추기게임이(가) 표시된 사진

자동 생성된 설명

A0=a+b

텍스트, 낱말맞추기게임이(가) 표시된 사진

자동 생성된 설명

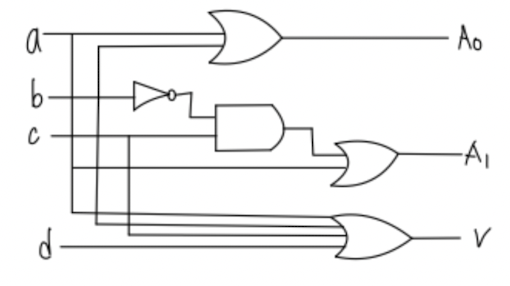
A1=a+b’c

텍스트, 쇼지, 낱말맞추기게임이(가) 표시된 사진

자동 생성된 설명

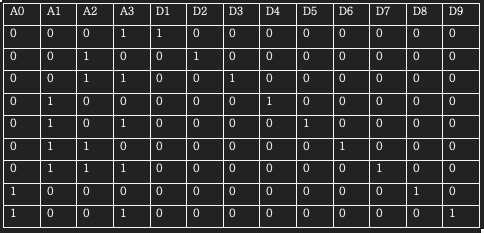
V=a+b+c+d

카르노맵을 통해 간소화한 식을 통해 4 to 2 priority encoder의 논리식을 구성하면 다음과 같다.



5.

Bcd to Decimal decoder는 4개의 입력으로 들어온 bcd값에 대한 decimal 값에 해당하는 출력값에 신호를 출력하는 회로이다. and gate로 decoder를 구성하엿으며, 진리표를 통해 bcd 값에 맞는 회선에 신호가 1로 출력되는 것을 볼 수 있다



진리표를 바탕으로 카르노맵을 구성한다.

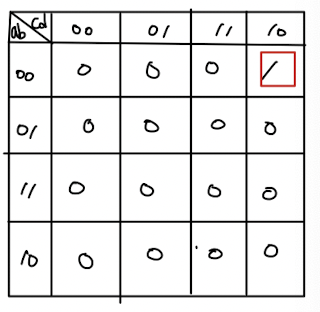
D1에 대한 카르노맵을 표현하면 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

D1=a’b’c’d

D2에 대한 카르노맵을 표현하면 다음과 같다.



D2=a’b’cd’

D3에 대한 카르노맵을 표현하면 다음과 같다.

낱말맞추기게임이(가) 표시된 사진

자동 생성된 설명

D3=a’b’cd

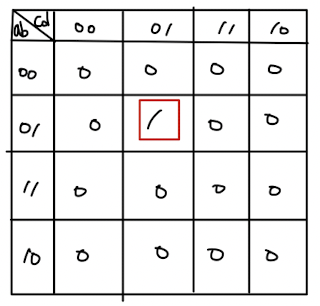
D4에 대한 카르노맵을 표현하면 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

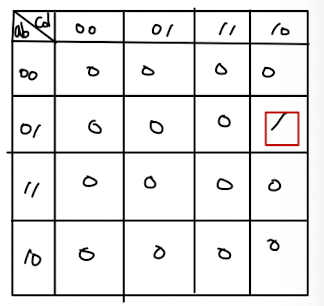
D4=a’bc’d’

D5에 대한 카르노맵을 표현하면 다음과 같다.



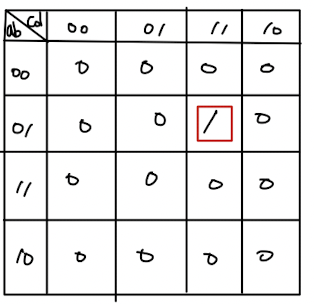
D5=a’bc’d

D6에 대한 카르노맵을 표현하면 다음과 같다.



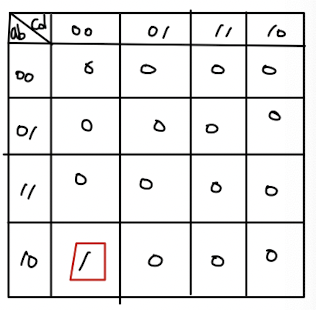
D6=a’bcd’

D7에 대한 카르노맵을 표현하면 다음과 같다.



D7=a’bcd

D8에 대한 카르노맵을 표현하면 다음과 같다.



D8=a’bc’d’

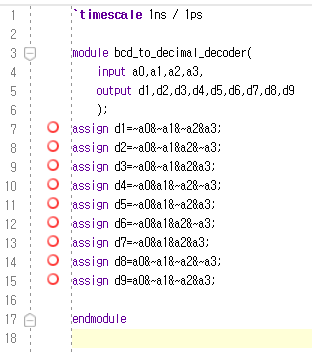
D9에 대한 카르노맵을 표현하면 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

D9=ab’c’d

카르노맵을 통해 간소화한 논리식을 베릴로그를 통해 구현하였다. 디자인 소스 파일에서 4개의 입력(a,b,c,d)와 9개의 출력(d1,d2,d3,d4,d5,d6,d7,d8,d9)을 선언하고, 카르노맵을 통해 구한 논리식을 각각 할당하였다.

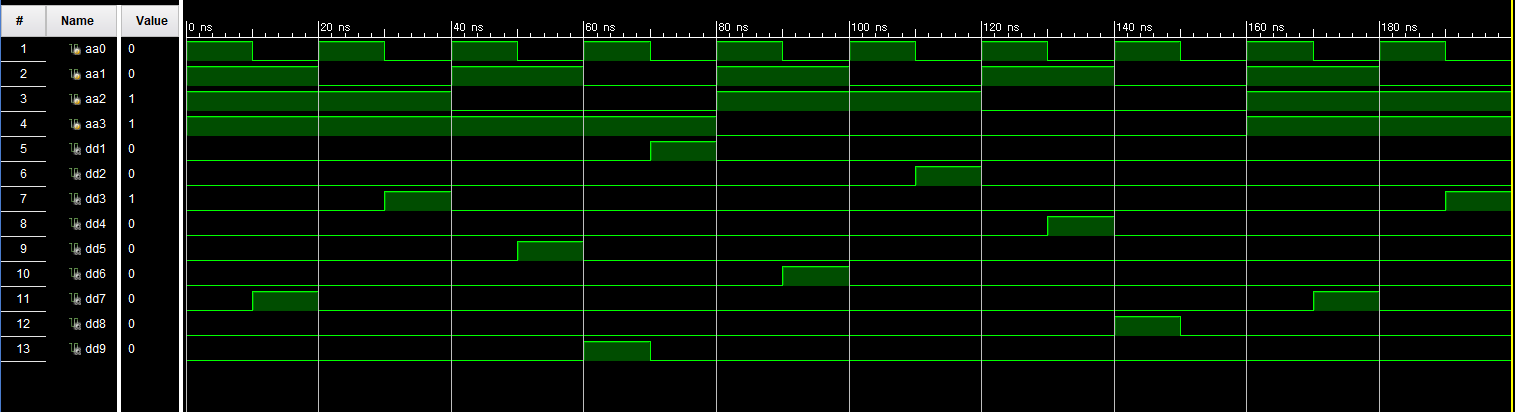


시뮬레이션 소스 파일을 다음과 같이 작성하였다.

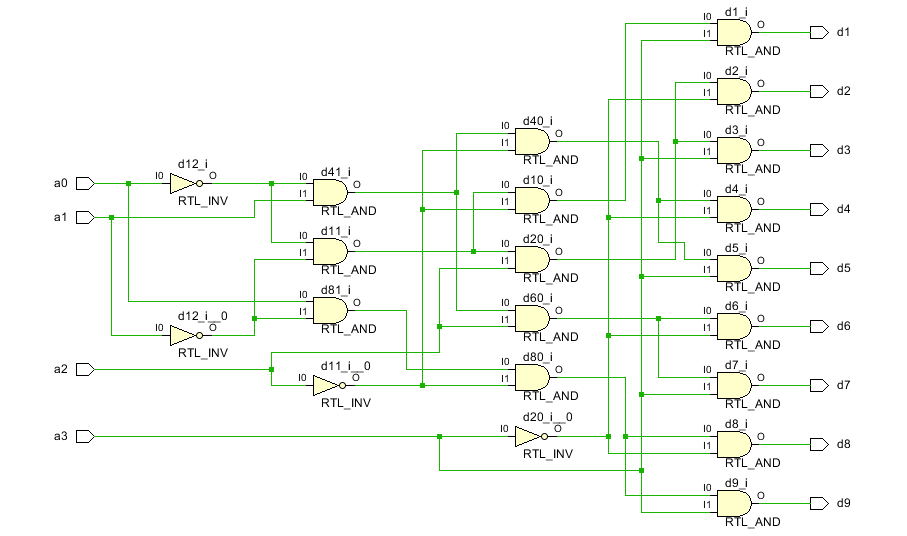
테이블이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 결과는 아래의 그림과 같으며, 입력값 aa,bb,cc,dd의 값에 따라 나타내는 bcd값에 대응하는 decimal값이 선택되어 출력되는 것을 확인할 수 있다.



Bcd to decimal decoder의 diagram을 Schematic으로 나타내었다.



6.

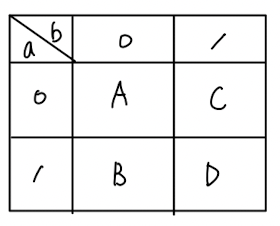
디코더와 인코더는 주로 정보를 전달하는 과정에서 응용된다. 정보를 전달할 때에는 디코더를 사용하여 정보를 디지털 신호로 변환한다. 반면 인코더는 디코더의 반대 역할로 정보를 받아서 다시 본래의 정보로 정보를 변환한다. 이러한 인코더와 디코더를 사용하는 정보 전달의 예시에는 이메일, 비디오, 마이크로 프로세서, 메모리 칩 등이 있다.

7.

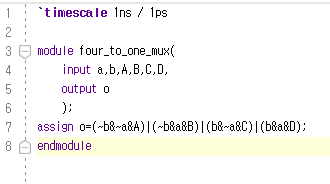
4 to 1 line mux는 6개의 입력(2개의 선택선과 4개의 정보)과 1개의 출력으로 구성되어, 선택선의 정보에 해당하는 정보를 출력하는 회로이다. 다음은 4 to 1 line mux의 진리표이다.



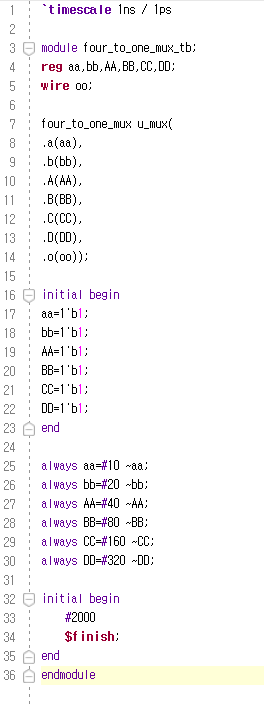
4 to 1 line mux를 카르노맵과 같이 표현하면 다음과 같다.



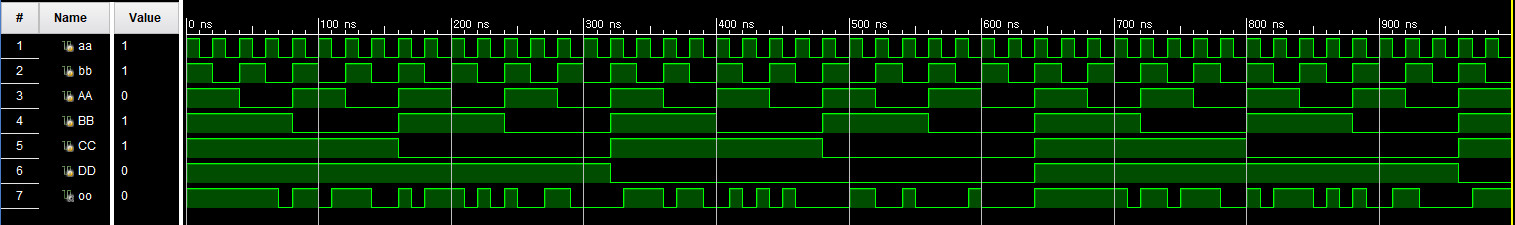
논리식을 베릴로그를 통해 구현하였다. 디자인 소스 파일에서 6개의 입력(입력선 a,b, 정보 A,B,C,D)와 1개의 출력(o)를 선언하고, 논리식을 다음과 같이 할당하였다.

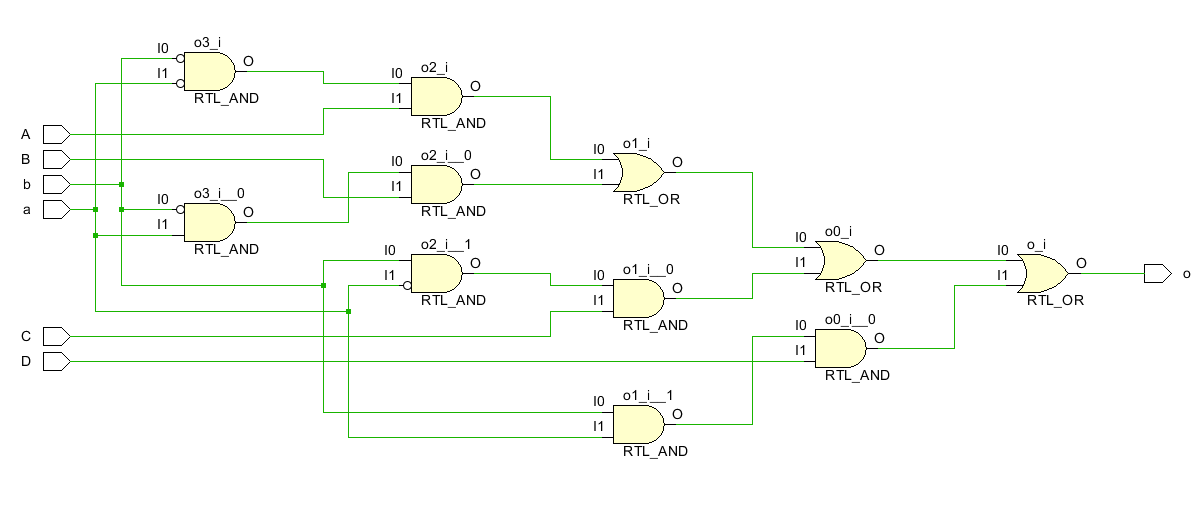


시뮬레이션 소스 파일을 다음과 같이 작성하였다.



시뮬레이션 결과는 아래의 그림과 같으며, 입력값 aa,bb의 값에 따라 대응하는 AA,BB,CC,DD값의 입력이 1이면 1을 출력하고 그렇지 않으면 0을 출력하는 것을 확인할 수 있다.

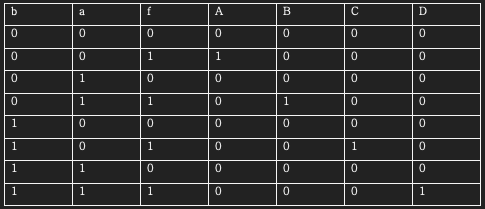




4 to 1 line mux의 diagram을 Schematic으로 나타내었다.

8.

1 to 4 line demux는 mux의 반대 역할을 하는 회로로서 3개의 입력(2개의 선택선과 1개의 데이터)와 4개의 출력으로 구성된다. 선택선에 따라 데이터가 1이면 선택선과 대응되는 출력이 1로 출력된다. 이러한 demux의 진리표를 나타내면 다음과 같다.



선택선 a,b에 따라 출력하고자 하는 값이 정해지는데, f 변수가 0이면 출력하고자 하는 값이 정해지더라도 활성화가 되지 않고, 1이면 해당 값이 1로 출력된다. 다음과 같이 논리식을 구성할 수 있다.

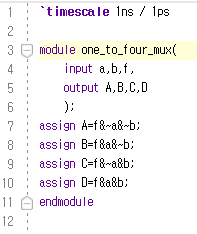
A=f&~a&~b

B=f&a&~b

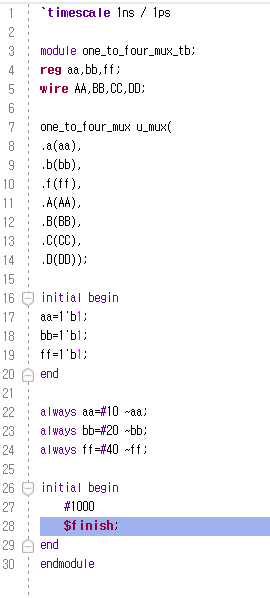
C=f&~a&b

D=f&a&b

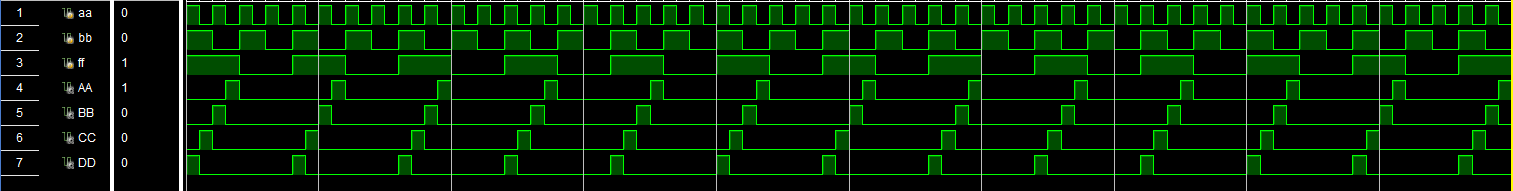
논리식을 베릴로그를 통해 구현하였다. 디자인 소스 파일에서 3개의 입력(입력선 a,b과 f)와 4개의 출력(A,B,C,D)를 선언하고, 논리식을 다음과 같이 할당하였다.

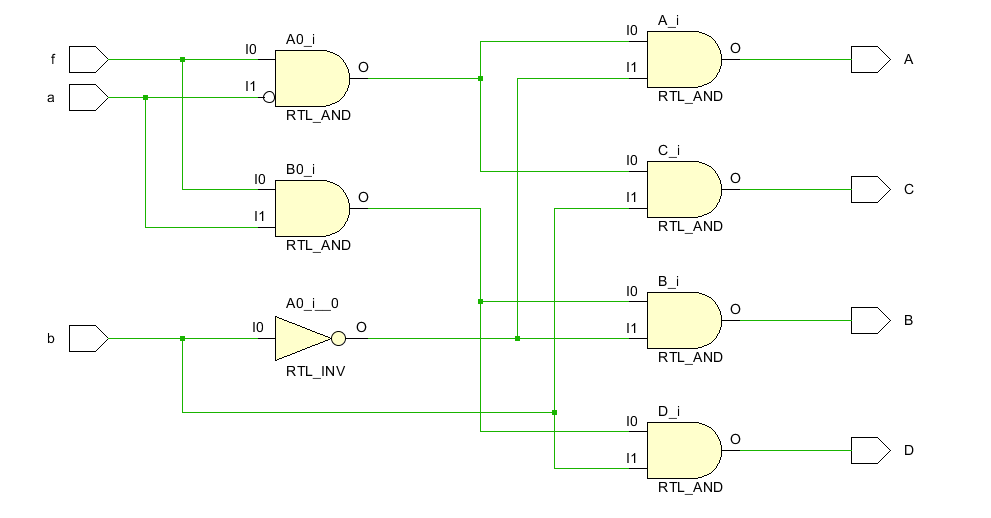


시뮬레이션 소스 파일을 다음과 같이 작성하였다.



시뮬레이션 결과는 아래의 그림과 같으며, 입력값 aa,bb의 값에 따라 ff가 1이면 해당 하는 출력변수가 1로 출력되고, ff가 0이면 0이 출력되는 것, 선택되지 않은 출력 변수는 0을 출력하는 것을 확인할 수 있다.

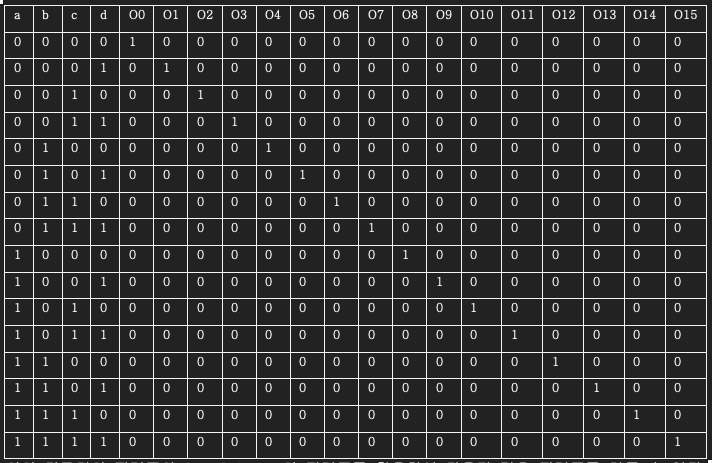




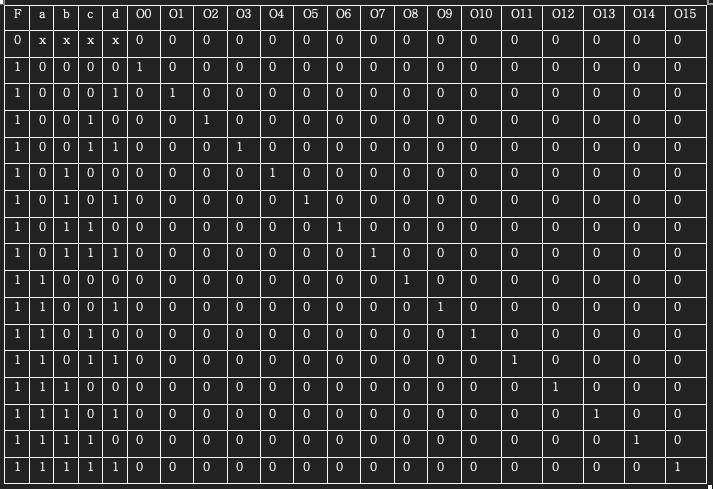
1 to 4 line demux의 diagram을 Schematic으로 나타내었다.

이러한 1 to 4 line demux를 활용하여 4 to 16 decoder를 다음과 같이 설계할 수 있다.

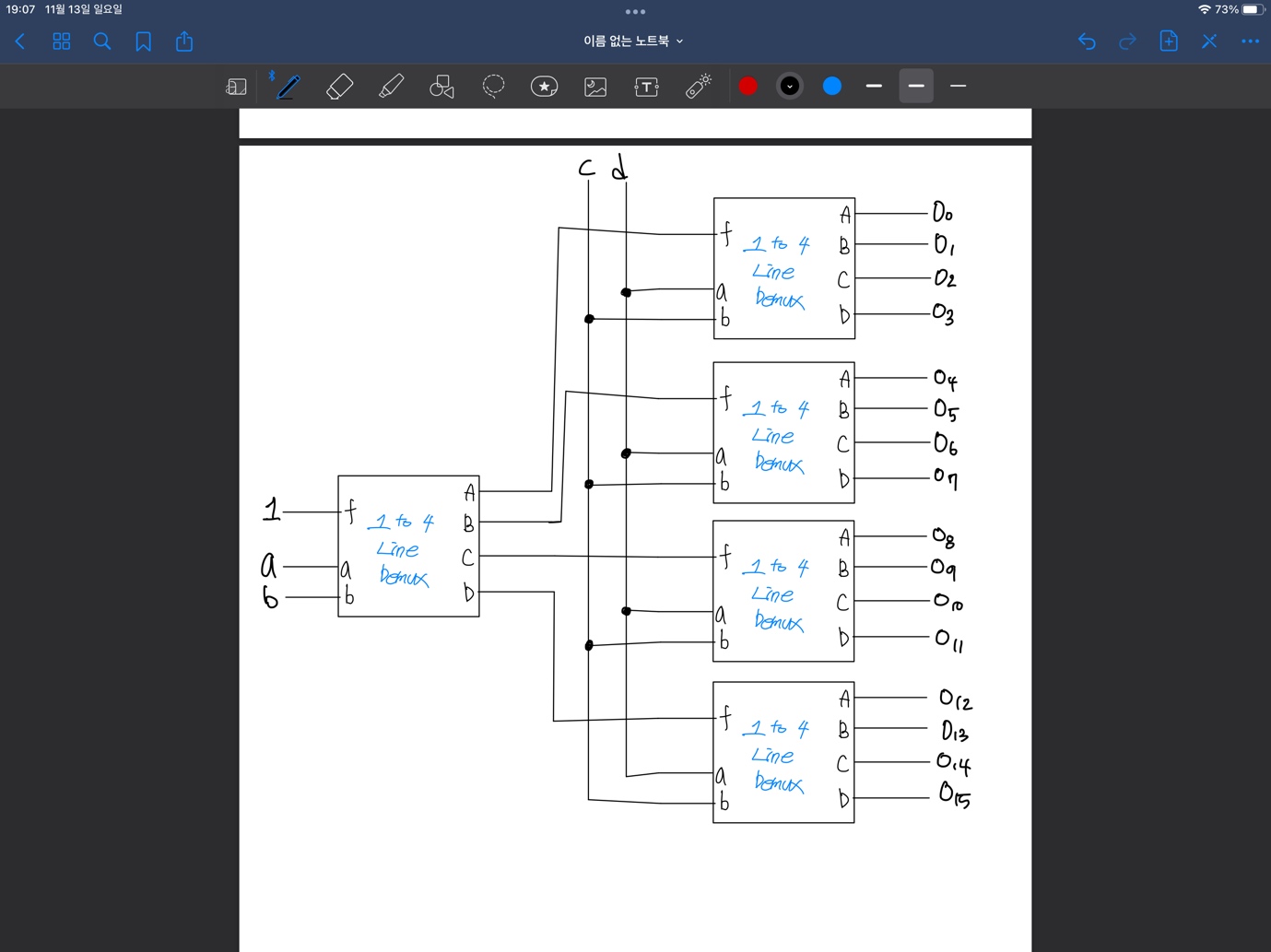
우선 4 to 16 디코더의 진리표는 다음과 같다.



위의 디코더의 진리표와 1 to 4 demux의 진리표를 활용하여 다음과 같은 진리표를 만들 수 있다.



Demux의 정보를 나타내는 f변수의 값이 1이면 디코더의 역할을 수행하도록, 0이면 디코더의 역할을 수행하지 않도록 한다. 또한 선택선 a,b,c,d를 활용하여 디코더가 어떤 인덱스에서 1을 출력할지 정한다. 이를 diagram으로 다음과 같이 나타낼 수 있다.



4 to 16 decoder를 1 to 4 demux를 활용하여 베릴로그에 표현하면 다음과 같다.

텍스트이(가) 표시된 사진

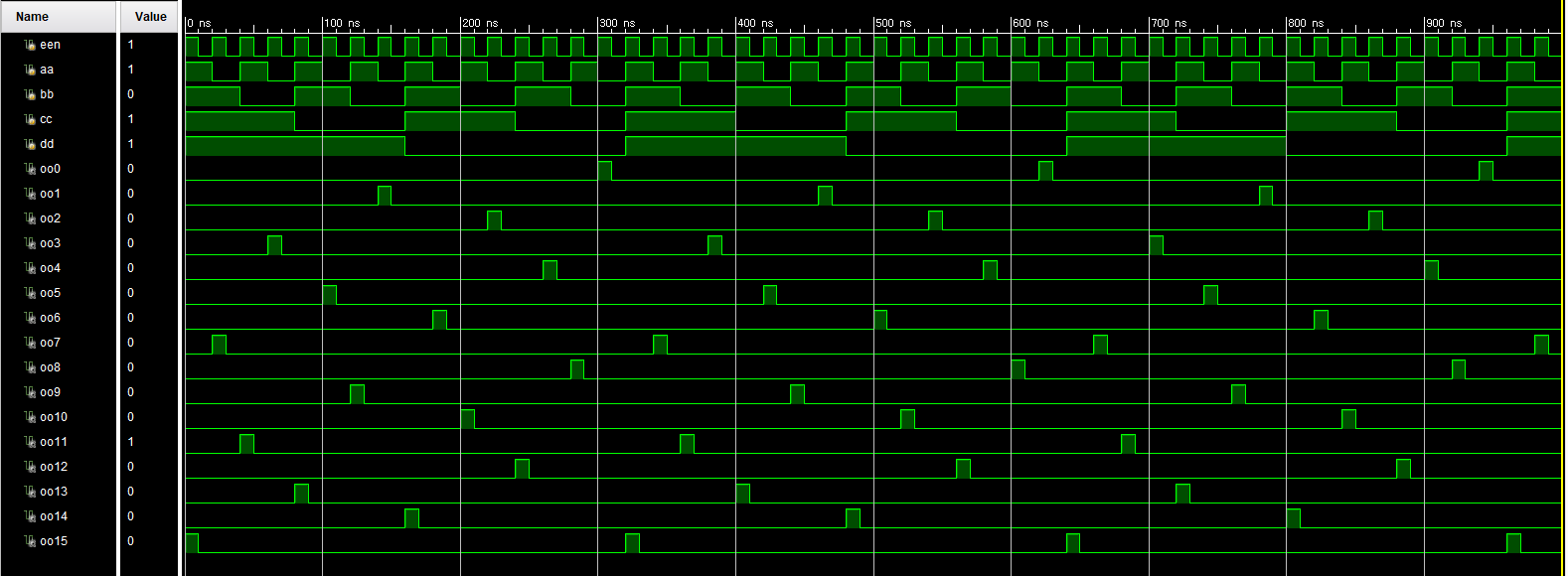
자동 생성된 설명

위 diagram을 코드로 표현하였다. 이에 대한 시뮬레이션 결과를 확인하기 위해 다음과 같이 시뮬레이션 코드를 작성하였다.

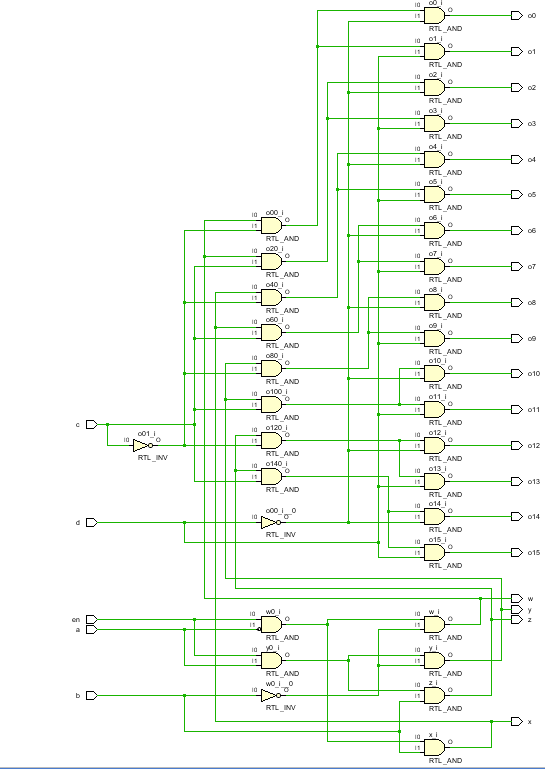
테이블이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 결과는 다음과 같다.



En값이 1이 될때에 디코더가 작동하여 aa,bb,cc,dd에 맞는 출력값이 1로 출력되고, 나머지 값은 0으로 출력되는 것을 알 수 있다.



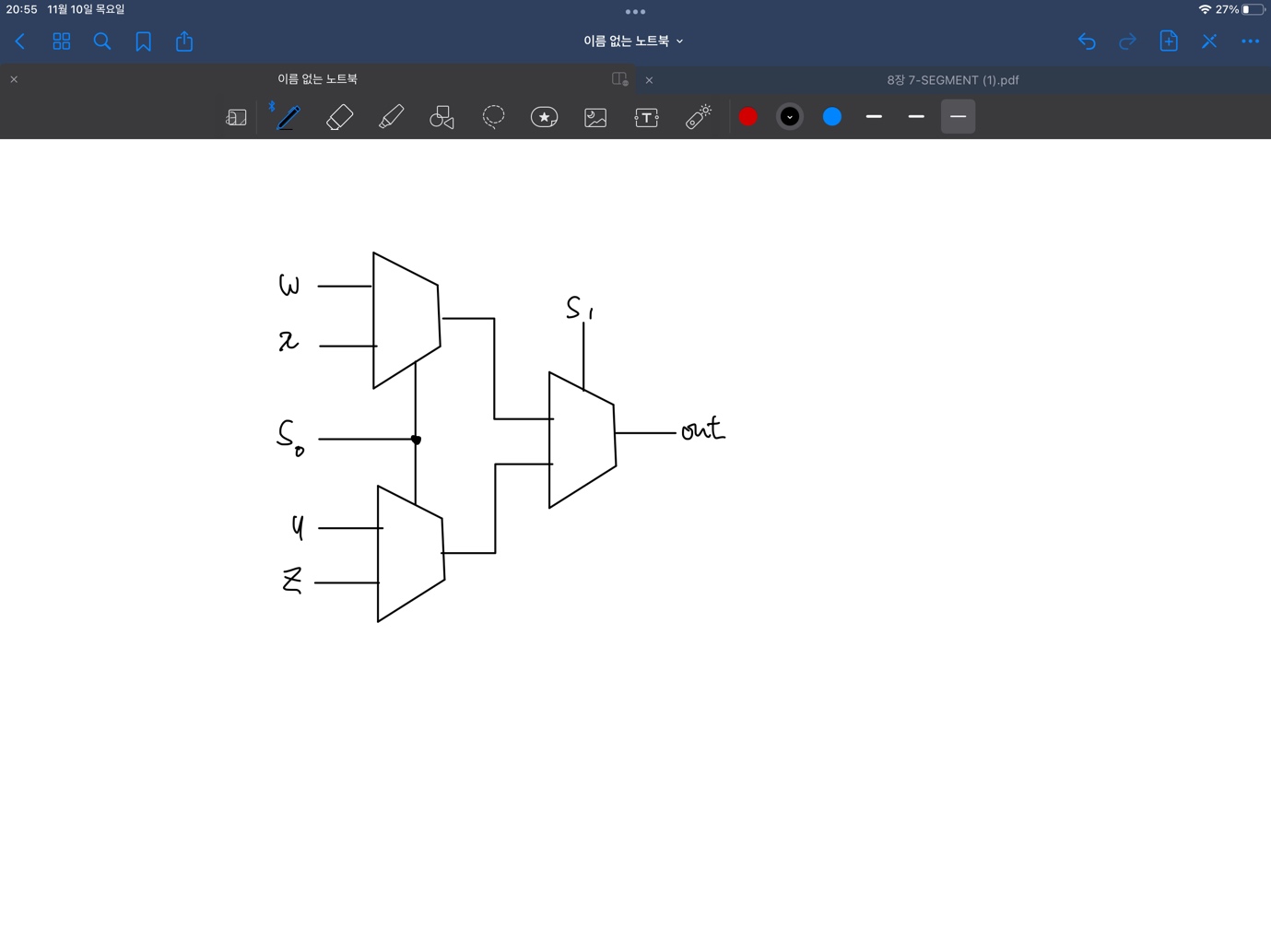
위를 schematic으로 나타내면 다음과 같다.

9.

Decoder, encoder, mux, demux를 구현하였다. 회로의 특성에 따라 진리표를 구현하고, 카르노맵을 활용하여 논리식을 작성하였다. 논리식을 베릴로그를 통해 구현하며, 시뮬레이션 및 FPGA를 통해 동작을 확인하였다. 디코더는 n개의 정보를 2^n개의 정보로 바꿔주고, 인코더는 디코더의 반대 역할을 수행하여 2^n개의 정보를 n개의 정보로 바꿔주었다. Mux는 2^n개의 입력선과 n개의 정보를 입력으로 받아, n개 중 하나의 정보를 선택하여 출력한다. Demux는 mux의 반대 역할을 하는 회로로 하나의 입력으로 받은 정보를 2^n개의 출력선 중 하나로 출력한다. 디코더와 인코더는 입력값에 따라 선택된 출력을 1로 출력하지만, mux와 demux는 입력값에 따라 선택된 출력을 출력한다는 차이를 보인다.

10.

2 to 4 디코더를 2 level로 연결하여 4 to 16 디코더를 만들 수 있듯이 2 to 1 mux를 2 level로 연결하여 4 to 1 mux를 만들 수 있다. 이러한 mux는 다음과 같이 구성한다.



2 level로 구현할 경우, 기존 2개의 입력선에 대한 출력값만을 가질 수 있던 2 To 1 mux가 더 많은 입력선에 대한 출력을 가능하도록 하여, 선택의 폭을 넓힐 수 있다.