6주차 결과보고서

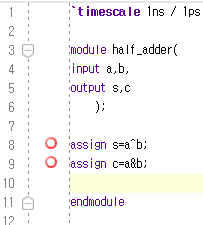
전공: 심리학과 학년: 3학년 학번: 20190345 이름: 김동현

**1.**

반 가산기, 전 가산기, 반 감산기, 전 감산기, 그리고 code converter를 이해하고 베릴로그를 통해 구현 및 시뮬레이션을 통해 확인한다. 이후 FPGA를 활용하여 값을 검증한다.

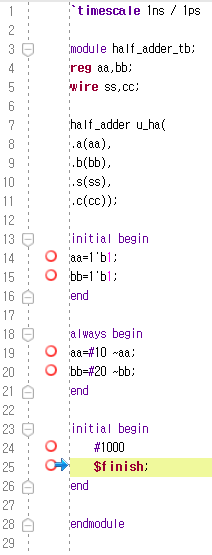
2.

반 가산기를 베릴로그를 통해 구현하였다.



<그림1. Half-adder design code>

위는 반 가산기의 design source code이다. 입력으로 더할 값 a,b를 선언하였고, 출력으로 자리올림 수(carry) c를, 합(sum) s를 선언하였다. 이후 s에는 a^b를, c에는 a&b를 할당하였다.



<그림2. Half-adder simulation code>

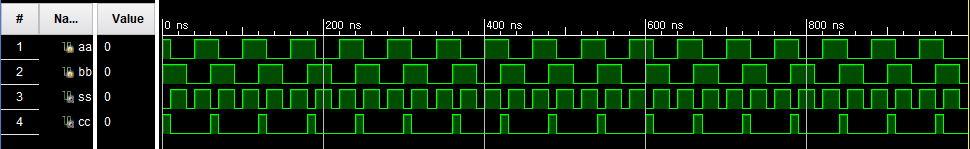
위는 반 가산기의 시뮬레이션 소스 코드이다.

테이블이(가) 표시된 사진

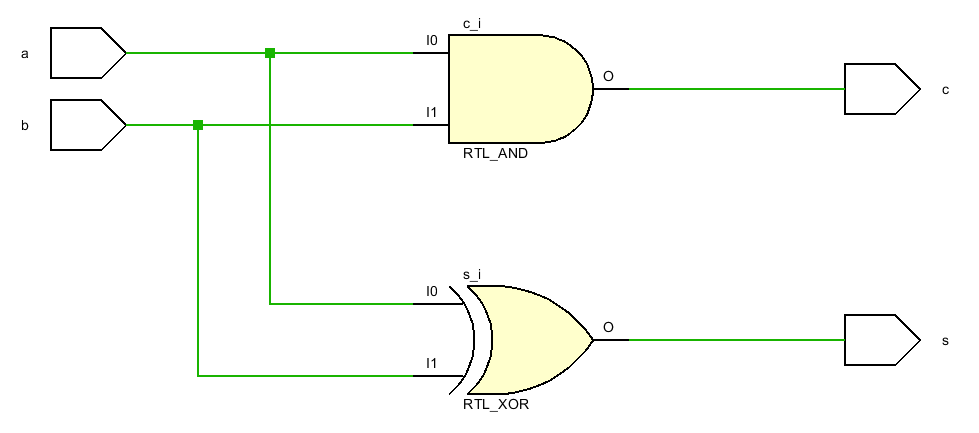
자동 생성된 설명

<그림3. Half-adder truth table>

위는 반 가산기의 입력값에 따른 출력값을 진리표로 나타내었다. S는 a,b의 1의 개수가 홀수일 때 1을, 짝수일때 0을 반환하는 xor연산을 나타내며, c는 a,b가 모두 1일 때 1을, 나머지 경우에 대해 0을 반환하는 and 연산을 나타내고 있다.

<그림4. Half-adder simulation>

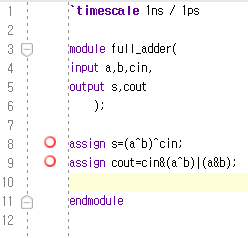
위 그림은 시뮬레이션 코드를 run하여 시뮬레이션을 실행한 결과이다. 진리표와 비교하였을 때, 값이 동일한 것을 확인할 수 있다.



<그림5. Half-adder schematics>

위는 반 가산기의 schematic이다. And와 xor gate를 활용하였다.

전 가산기를 베릴로그를 통해 구현하였다.



<그림6. full-adder design code>

위는 전 가산기의 design source code이다. 입력으로 더할 값 a,b와 이전 연산의 자리올림수cin를 선언하였고, 출력으로 자리올림 수(carry) cout를, 합(sum) s를 선언하였다. 이후 s에는 (a^b)^cin를, cout에는 cin&(a^b)|(a&b)를 할당하였다.

테이블이(가) 표시된 사진

자동 생성된 설명

<그림7. full-adder simulation code>

위는 전 가산기의 시뮬레이션 소스 코드이다.

테이블이(가) 표시된 사진

자동 생성된 설명

<그림8. full-adder truth table>

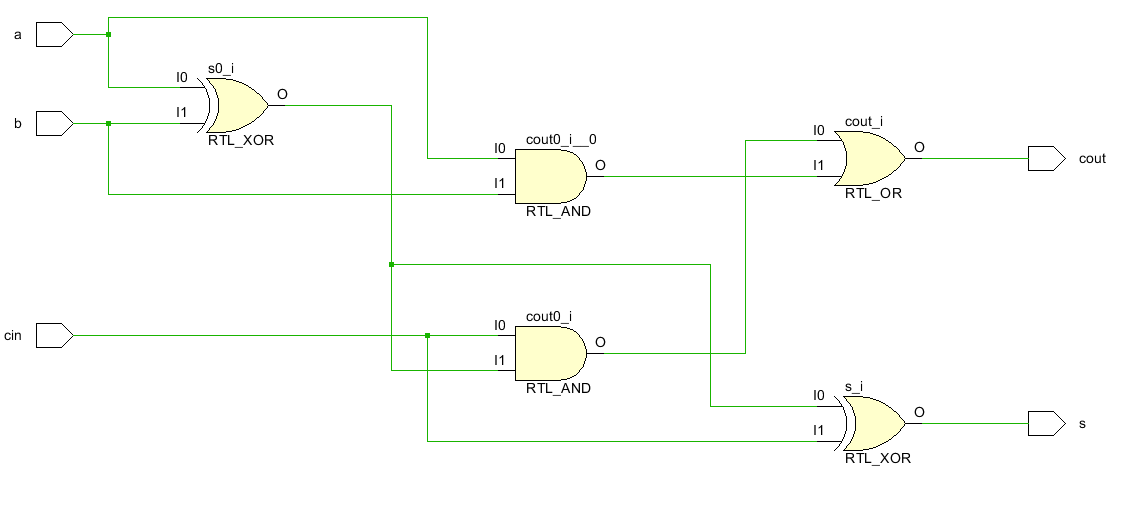
위는 전 가산기의 입력값에 따른 출력값을 진리표로 나타내었다.

텍스트, 실내, 키보드이(가) 표시된 사진

자동 생성된 설명

<그림9. full-adder simulation>

위 그림은 시뮬레이션 코드를 run하여 시뮬레이션을 실행한 결과이다. 진리표와 비교하였을 때, 값이 동일한 것을 확인할 수 있다.

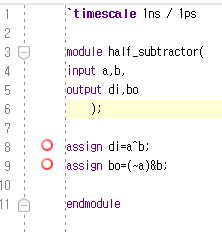


<그림10. full-adder schematics>

위는 반 가산기의 schematic이다. And와 xor gate 2개 or gate 1개를 활용하였다.

3. 전 감산기 및 반 감산기의 시뮬레이션 결과 및 과정에 대해 설명

반 감산기를 베릴로그를 통해 구현하였다.



<그림11. Half-subtractor design code>

위는 반 감산기의 design source code이다. 입력으로 뺄 값 a,b를 선언하였고, 출력으로 빌려올 갑(borrow) bo를, 차(difference) di를 선언하였다. 이후 di에는 a^b를, bo에는 (~a)&b를 할당하였다.

테이블이(가) 표시된 사진

자동 생성된 설명

<그림12. Half-subtractor simulation code>

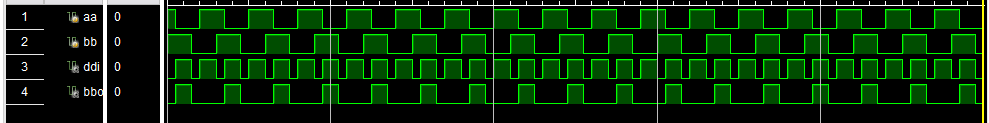
위는 반 감산기의 시뮬레이션 소스 코드이다.

테이블이(가) 표시된 사진

자동 생성된 설명

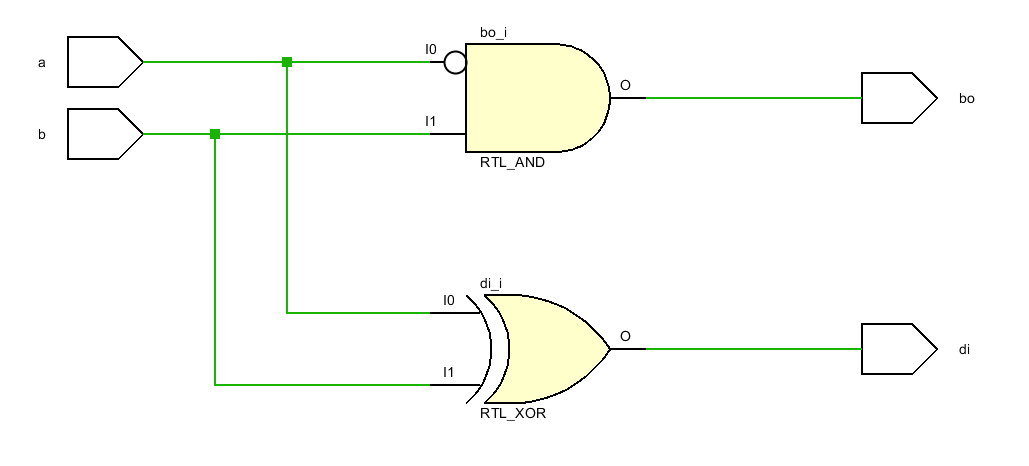
<그림13. Half-subtractor truth table>

위는 반 감산기의 입력값에 따른 출력값을 진리표로 나타내었다. Di는 a-b의 값을 나타내며, 0-1=-1의 경우에는 bo값을 1로 바꿔서 값을 빌려와 빼기를 수행한다.



<그림14. Half-subtractor simulation>

위 그림은 시뮬레이션 코드를 run하여 시뮬레이션을 실행한 결과이다. 진리표와 비교하였을 때, 값이 동일한 것을 확인할 수 있다.



<그림15. Half-subtractor schematics>

위는 반 감산기의 schematic이다. And와 xor, not gate를 활용하였다.

전 감산기를 베릴로그를 통해 구현하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림16. full-subtractor design code>

위는 전 감산기의 design source code이다. 입력으로 더할 값 a,b와 이전 연산이 값을 빌려가는지 여부를 나타내는 bin를 선언하였고, 출력으로 다음 연산에서 빌려오는 지를 나타내는 수(borrow) bout를, 차(difference) dout를 선언하였다. 이후 bout에는 (a^b)^bin|(~a)&b를, dout에는 a^b^bin를 할당하였다.

테이블이(가) 표시된 사진

자동 생성된 설명

<그림17. full-subtractor simulation code>

위는 전 감산기의 시뮬레이션 소스 코드이다.

테이블이(가) 표시된 사진

자동 생성된 설명

<그림18. full-subtractor truth table>

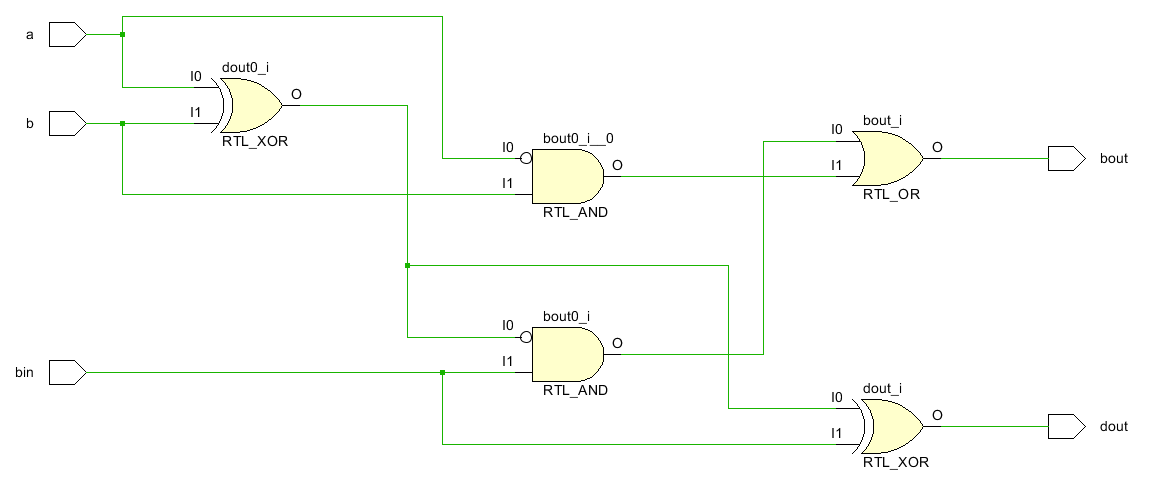
위는 전 감산기의 입력값에 따른 출력값을 진리표로 나타내었다.

텍스트, 키보드, 실내이(가) 표시된 사진

자동 생성된 설명

<그림19. full-subtractor simulation>

위 그림은 시뮬레이션 코드를 run하여 시뮬레이션을 실행한 결과이다. 진리표와 비교하였을 때, 값이 동일한 것을 확인할 수 있다.



<그림20. full-subtractor schematics>

위는 반 감산기의 schematic이다. And와 xor, not gate 2개 or gate 1개를 활용하였다.

4.

테이블이(가) 표시된 사진

자동 생성된 설명

<그림21. 10진수, 8421 code, 2421 code>

10진수 decimal에 대한 8421 code와 2421 code에 대한 진리표이다. A,b,c,d는 8421 코드에 대한 자리수를 나타내고, e,f,g,h는 2421 코드에 대한 자리수를 나타낸다.

위의 표를 이용하여, 8421코드를 2421로 바꾸는 code converter에 대한 카르노맵을 구성하였다.

2421 code에 대한 변수 e,f,g,h는 sum of product를 활용하여 나타내고 있다. 추가적으로 변수e2,f2,g2,h2를 추가하여 product of sum을 활용하여 나타냈다.

테이블이(가) 표시된 사진

자동 생성된 설명

<그림22. K-map of variable e>

위 표는 변수 e에 대한 카르노 맵이다. 해당 맵을 간소화하면

E=a+db+cb

E2=(a+b)(a+c+d)로 표현할 수 있다.

테이블이(가) 표시된 사진

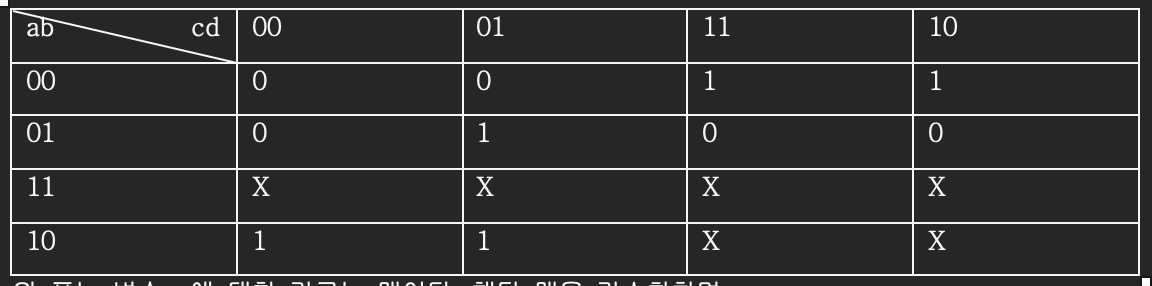
자동 생성된 설명

<그림23. K-map of variable f>

위 표는 변수 f에 대한 카르노 맵이다. 해당 맵을 간소화하면

f=a+bc+bd’

f2=(a+b)(a+c+d’)로 표현할 수 있다.



<그림24. K-map of variable g>

위 표는 변수 g에 대한 카르노 맵이다. 해당 맵을 간소화하면

g=a+bc’d+b’c

g2=(a+b+c)(a+c+d)(b’+c’)로 표현할 수 있다.

텍스트, 클래퍼보드이(가) 표시된 사진

자동 생성된 설명

<그림25. K-map of variable h>

위 표는 변수 h에 대한 카르노 맵이다. 해당 맵을 간소화하면

h=d

h2=d로 표현할 수 있다.

위 카르노맵을 바탕으로 베릴로그 내에서 코드를 구현하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림26. Half-subtractor design code>

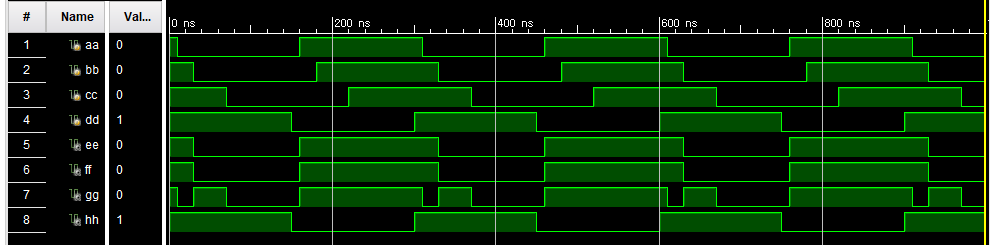
위는 code converter의 design source code이다. 입력으로 8421 code a,b,c,d를 선언하였고, 출력으로 2421 code e,f,g,h를 선언하였다. 이후 카르노맵에서 구한 식을 대입하여 할당하였다.

테이블이(가) 표시된 사진

자동 생성된 설명

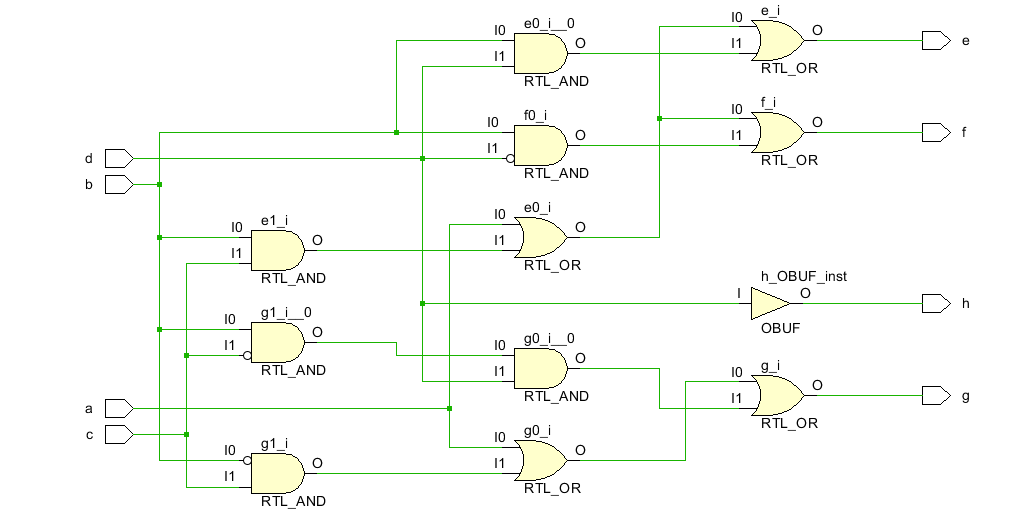
<그림27. Half-subtractor simulation code>

위는 code converter의 시뮬레이션 소스 코드이다.



<그림28. Half-subtractor simulation>

위 그림은 시뮬레이션 코드를 run하여 시뮬레이션을 실행한 결과이다. 위의 진리표와 비교하였을 때, 값이 동일한 것을 확인할 수 있다.



<그림29. Half-subtractor schematics>

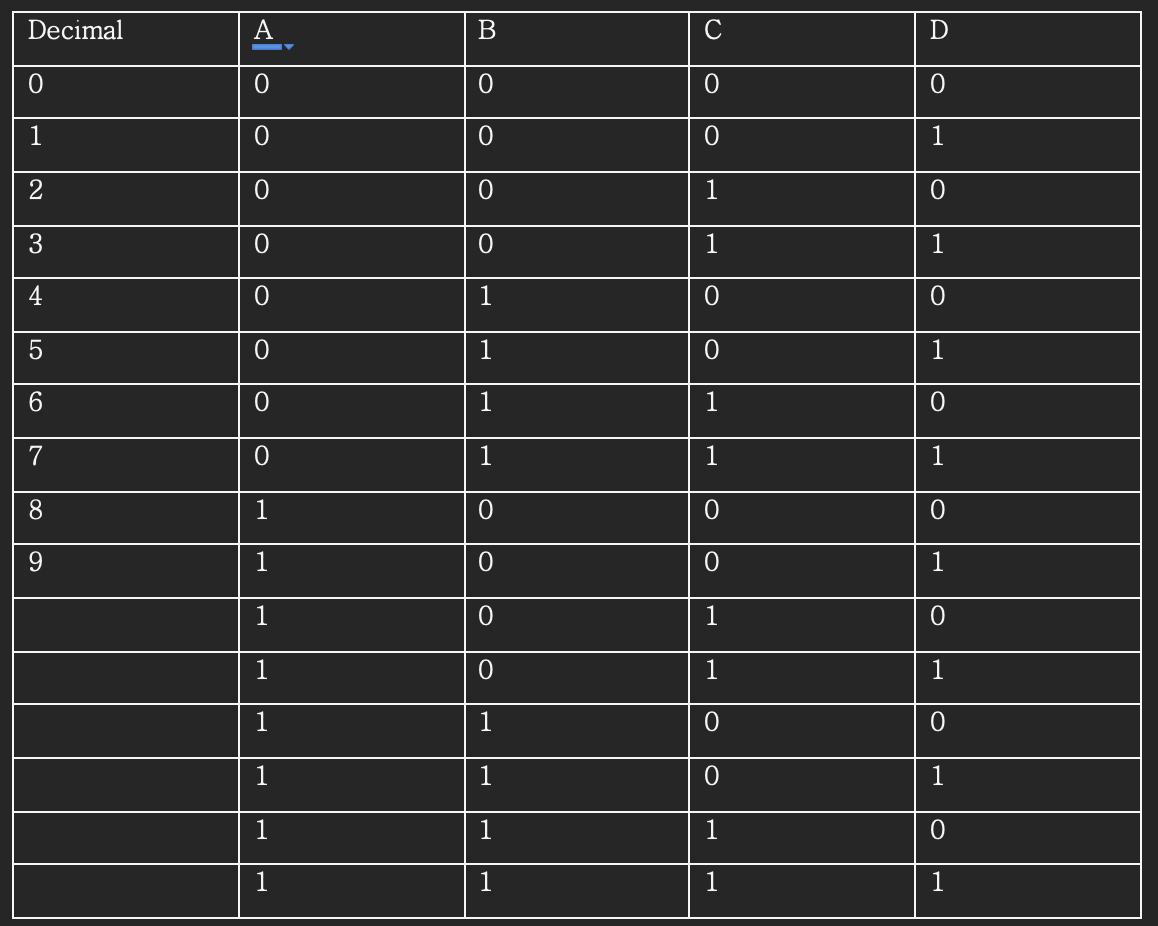
위는 code converter의 schematic이다.

5.

반 가산기, 전 가산기, 반 감산기, 전 감산기, code converter를 베릴로그를 활용해 구현하였다. 진리표 및 카르노맵을 통해 식을 구성하고, design source code를 구현하였다. 이후 simulation을 활용하여 진리표와 비교를 하였다. 반 가산기는 덧셈 연산을 하며, 입력값의 합과 자리올림수를 반환한다. 전가산기는 반가산기 2개와 or gate를 활용하여, 덧셈 연산을 수행한다. 반 감산기는 뺄셈 연산을 하며, 입력값의 차와 수를 빌려오는지 여부를 반환한다. 전 감산기는 반 감산기 2개를 활용하여, 뺄셈 연산을 수행한다. Code converter를 구현하기 위해, 카르노맵을 구성하였다. 카르노맵을 구현하여 이를 sum of product와 product of sum으로 간소화하여 포함하였다.

6.

BCD code는 4비트의 이진수를 활용하여 0부터 9까지를 표현한다.



<그림30. Truth table of BCD code>

다음과 같이 0000(2)부터 1001(2)까지의 이진법을 사용하여 10개의 정수를 표현하고 있다. 이러한 BCD 코드는 자리수 마다 4비트 단위로 구성하여 구현이 편리하다는 장점이 있지만, 1010(2)부터 1111(2)까지 6개를 낭비하고 있다는 단점이 있다.

7. 참고문헌

컴퓨터공학실험2 교재

Alan B. Marcvitz, Introduction to Logic Design, *McGraw-Hill*, 2010