9주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20211533 이름: 남정연

**1.**

2 to 4 Decoder는 2 bit 입력을 받아서 각각의 경우에 해당하는 4개의 출력을 내보내는 회로로, 아래와 같은 진리표를 가진다.

AND

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | d0 | d1 | d2 | d3 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

NAND

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | d0 | d1 | d2 | d3 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |

카르노 맵 및 SOP는 아래와 같다.

d0

AND

|  |  |  |
| --- | --- | --- |
| a|b | 0 | 1 |
| 0 | 0 | 0 |
| 1` | 0 | 1 |

NAND

|  |  |  |
| --- | --- | --- |
| a|b | 0 | 1 |
| 0 | 1 | 1 |
| 1` | 1 | 0 |

d1

AND

|  |  |  |
| --- | --- | --- |
| a|b | 0 | 1 |
| 0 | 0 | 0 |
| 1` | 1 | 0 |

NAND

|  |  |  |
| --- | --- | --- |
| a|b | 0 | 1 |
| 0 | 1 | 1 |
| 1` | 0 | 1 |

d2

AND

|  |  |  |
| --- | --- | --- |
| a|b | 0 | 1 |
| 0 | 0 | 1 |
| 1` | 0 | 0 |

NAND

|  |  |  |
| --- | --- | --- |
| a|b | 0 | 1 |
| 0 | 1 | 0 |
| 1` | 1 | 1 |

d3

AND

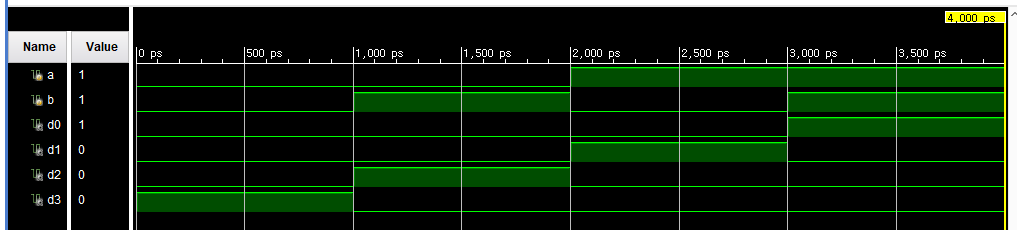
|  |  |  |
| --- | --- | --- |
| a|b | 0 | 1 |
| 0 | 1 | 0 |
| 1` | 0 | 0 |

NAND

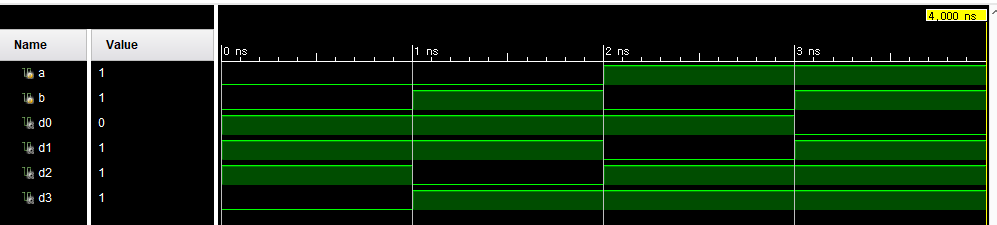
|  |  |  |
| --- | --- | --- |
| a|b | 0 | 1 |
| 0 | 0 | 1 |
| 1` | 1 | 1 |

위 카르노맵을 바탕으로 시뮬레이션을 수행하면 아래와 같은 결과가 나타난다.

AND



NAND



**2.**

4 to 2 Encoder는 4 bit 입력을 받아서 각각의 경우에 해당하는 2개의 출력을 내보내는 회로로, 아래와 같은 진리표를 가진다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | e0 | e1 |
| 0 | 0 | 0 | 0 | x | x |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | x | x |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | x | x |
| 0 | 1 | 1 | 0 | x | x |
| 0 | 1 | 1 | 1 | x | x |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | x | x |
| 1 | 0 | 1 | 0 | x | x |
| 1 | 0 | 1 | 1 | x | x |
| 1 | 1 | 0 | 0 | x | x |
| 1 | 1 | 0 | 1 | x | x |
| 1 | 1 | 1 | 0 | x | x |
| 1 | 1 | 1 | 1 | x | x |

위 진리표를 바탕으로 카르노맵은 아래와 같이 그려진다.

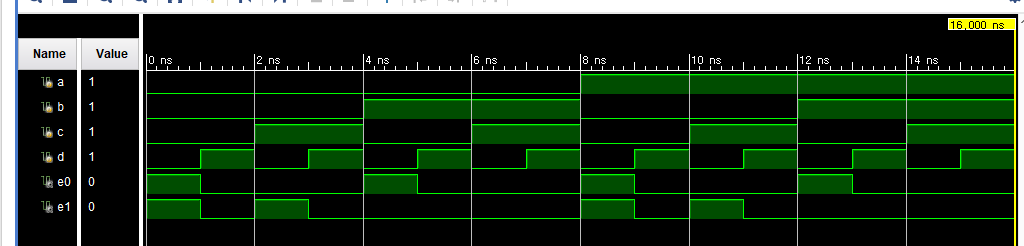
e0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | x | 0 | x | 0 |
| 01 | 1 | x | x | x |
| 11 | x | x | x | x |
| 10 | 1 | x | x | x |

e1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | x | 0 | x | 1 |
| 01 | 0 | x | x | x |
| 11 | x | x | x | x |
| 10 | 1 | x | x | x |

위 카르노맵을 바탕으로 아래와 같은 시뮬레이션 결과가 도출된다.



**3.**

허용되는 입력 형태(0001, 0010, 0100, 1000)를 제외한 나머지 입력 형태는 don’t care를 의미한다.

**4.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | e0 | e1 |
| 0 | 0 | 0 | 0 | x | x |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

위 진리표를 바탕으로 카르노맵은 아래와 같이 그려진다.

e0 (SOP = b+a)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | x | 0 | x | 0 |
| 01 | 1 | x | x | x |
| 11 | x | x | x | x |
| 10 | 1 | x | x | x |

e1 (SOP=b’+a)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | x | 0 | x | 1 |
| 01 | 0 | x | x | x |
| 11 | x | x | x | x |
| 10 | 1 | x | x | x |

위 카르노맵과 도출된 SOP에 따라 e0=a+b, e1=a+b’의 논리 회로를 구성할 수 있다.

**5.**

BCD to Decimal decoder는 4bit 입력 BCD 값에 따라 1~9의 각 숫자를 나타내는 출력 4개를 도출하는 회로로 아래와 같은 진리표를 가진다.

빈 칸은 0을 의미한다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | o1 | o2 | o3 | o4 | o5 | o6 | o7 | o8 | o9 |
| 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |
| 0 | 0 | 0 | 1 |  |  |  |  |  |  |  | 1 |  |
| 0 | 0 | 1 | 0 |  |  |  | 1 |  |  |  |  |  |
| 0 | 0 | 1 | 1 |  |  |  |  |  |  |  |  |  |
| 0 | 1 | 0 | 0 |  | 1 |  |  |  |  |  |  |  |
| 0 | 1 | 0 | 1 |  |  |  |  |  |  |  |  |  |
| 0 | 1 | 1 | 0 |  |  |  |  |  | 1 |  |  |  |
| 0 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |
| 1 | 0 | 0 | 0 | 1 |  |  |  |  |  |  |  |  |
| 1 | 0 | 0 | 1 |  |  |  |  |  |  |  |  | 1 |
| 1 | 0 | 1 | 0 |  |  |  |  | 1 |  |  |  |  |
| 1 | 0 | 1 | 1 |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 0 | 0 |  |  | 1 |  |  |  |  |  |  |
| 1 | 1 | 0 | 1 |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 0 |  |  |  |  |  |  | 1 |  |  |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |

위 진리표를 바탕으로 아래와 같은 카르노맵이 그려진다. (빈 칸은 0을 의미한다)

o1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab|cd | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 |  |  |  |  |
| 11 |  |  |  |  |
| 10 | 1 |  |  |  |

o2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab|cd | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 | 1 |  |  |  |
| 11 |  |  |  |  |
| 10 |  |  |  |  |

o3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab|cd | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 |  |  |  |  |
| 11 | 1 |  |  |  |
| 10 |  |  |  |  |

o4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab|cd | 00 | 01 | 11 | 10 |
| 00 |  |  |  | 1 |
| 01 |  |  |  |  |
| 11 |  |  |  |  |
| 10 |  |  |  |  |

o5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab|cd | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 |  |  |  |  |
| 11 |  |  |  |  |
| 10 |  |  |  | 1 |

o6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab|cd | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 |  |  |  | 1 |
| 11 |  |  |  |  |
| 10 |  |  |  |  |

o7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab|cd | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 |  |  |  |  |
| 11 |  |  |  | 1 |
| 10 |  |  |  |  |

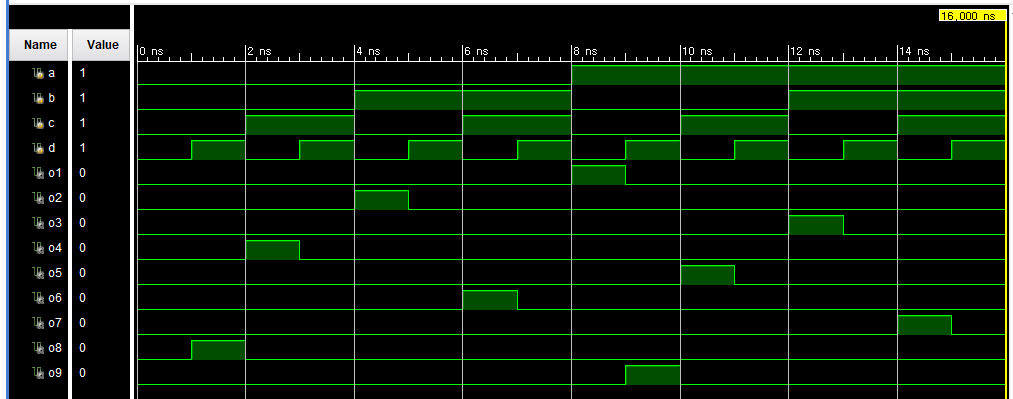
o8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab|cd | 00 | 01 | 11 | 10 |
| 00 |  | 1 |  |  |
| 01 |  |  |  |  |
| 11 |  |  |  |  |
| 10 |  |  |  |  |

o9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab|cd | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 |  |  |  |  |
| 11 |  |  |  |  |
| 10 |  | 1 |  |  |

위 카르노맵을 바탕으로 아래와 같은 시뮬레이션 결과가 도출되었다.



**6.**

인코더와 디코더를 응용하여 BCD Encoder 및 BCD Decoder를 만들어낼 수 있다. BCD Encoder의 경우에는 0~9까지의 Decimal Input 10개를 받아 각각의 해당하는 수를 1,2,4,8의 해당하는 4개의 output으로 출력하는 인코더이며, Decoder는 이의 역과정으로 5번에서 수행한 것처럼 1,2,4,8에 해당하는 4개의 input에 대하여 0~9까지 10개의 출력으로 각각에 해당하는 수를 도출하는 회로이다.

**7.**

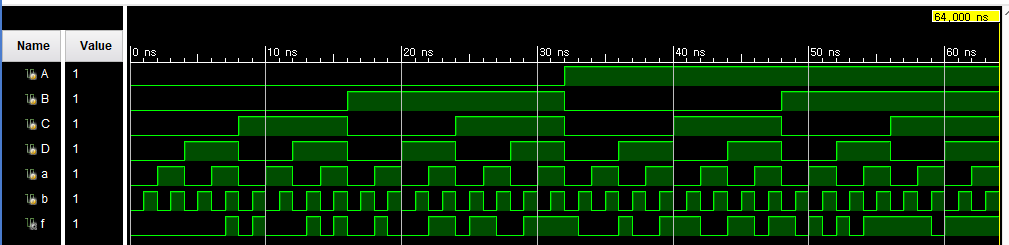
4 to 1 MUX는 4개의 입력 및 2개의 셀렉터에 대하여 하나의 출력을 내보내는 회로이다. 아래와 같은 코드 및 진리표가 나타난다.

CODE

f = (A & ~b & ~a) | (B & ~b & a) | (C & b & ~a) | (D & b & a);

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | a | b | f |
| 1 | x | x | x | 0 | 0 | 1 |
| 0 | x | x | x | 0 | 0 | 0 |
| x | x | 1 | x | 0 | 1 | 1 |
| x | x | 0 | x | 0 | 1 | 0 |
| x | 1 | x | x | 1 | 0 | 1 |
| x | 0 | x | x | 1 | 0 | 0 |
| x | x | x | 1 | 1 | 1 | 1 |
| x | x | x | 0 | 1 | 1 | 0 |

위 진리표를 바탕으로 아래와 같은 시뮬레이션 결과가 도출되었다.



**8.**

1 to 4 deMUX는 1개의 입력 및 2개의 셀렉터에 대하여 4개의 출력을 내보내는 회로로, 아래와 같은 코드 및 진리표가 나타난다.

CODE

assign A = f & ~a & ~b;

assign B = f & a & ~b;

assign C = f & ~a & b;

assign D = f & a & b;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| f | a | b | A | B | C | D |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | x | x | 0 | 0 | 0 | 0 |

위 진리표를 바탕으로 아래와 같은 시뮬레이션 결과가 도출되었다.

텍스트, 시계, 점수판이(가) 표시된 사진

자동 생성된 설명

**9.**

모든 실험에 대해 각각의 진리표 및 카르노맵, 코드에 대한 정확한 결과가 시뮬레이션을 통해 도출되었고 이에 따라 각 실험이 정확하고 정밀하게 진행되었음을 알 수 있다.

**10.**

BCD Adder는 Decimal Number의 덧셈 연산을 바로 수행하기 위해 만들어지 회로로 이진 코드에 해당하는 입력 2개(총 8개)를 받아 내부적으로 덧셈을 수행하고 그 결과를 BCD 코드로 변환하여 결과적으로는 BCD 코드를 출력하는 회로를 말한다.