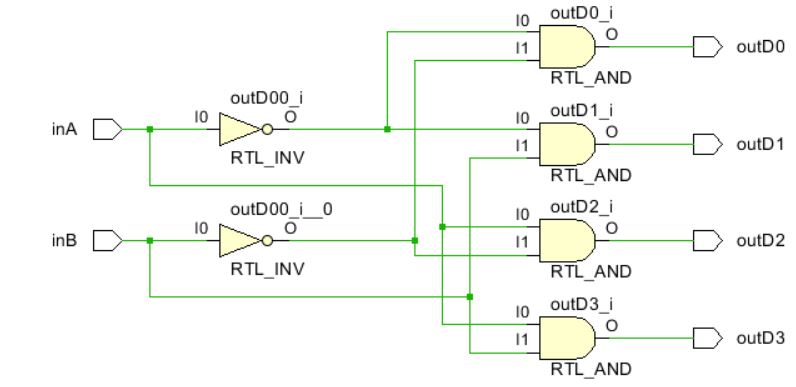
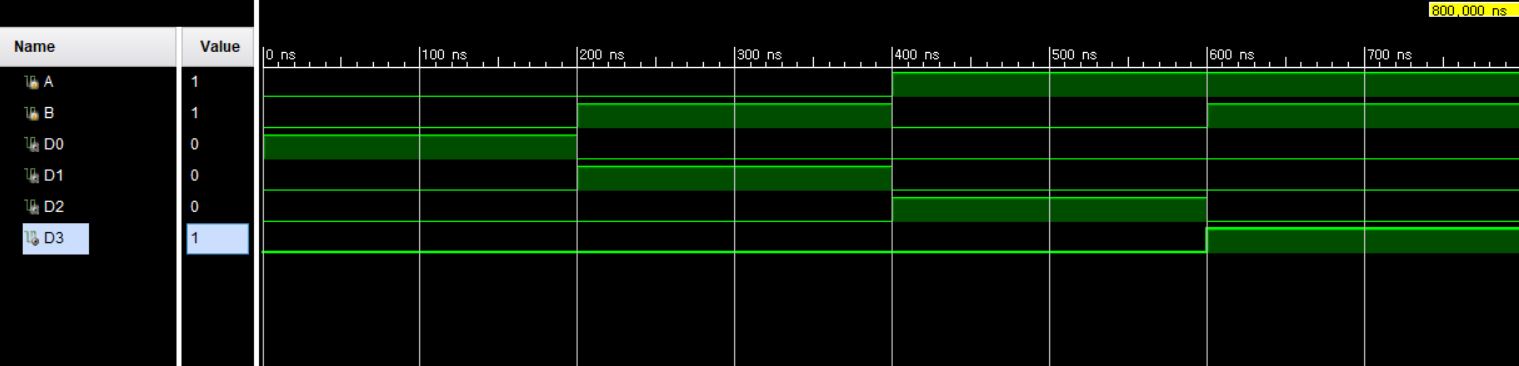
9주차 결과보고서

전공: 물리학과 학년: 3학년 학번: 20191286 이름: 김나현

1. 2 to 4 Decoder의 결과 및 simulation 과정에 대해서 설명하시오(Truth table 작성 및 k-map 포함)(AND, NAND 게이트).

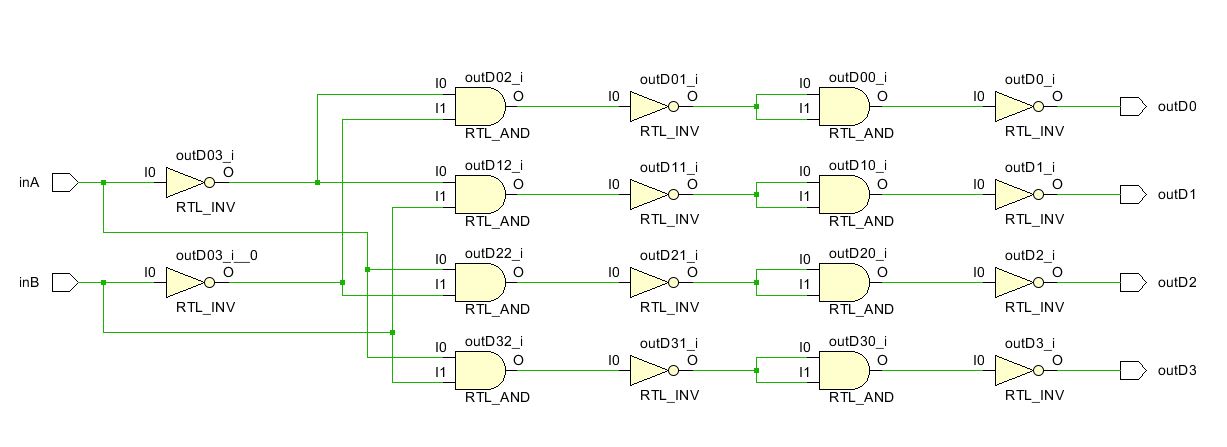


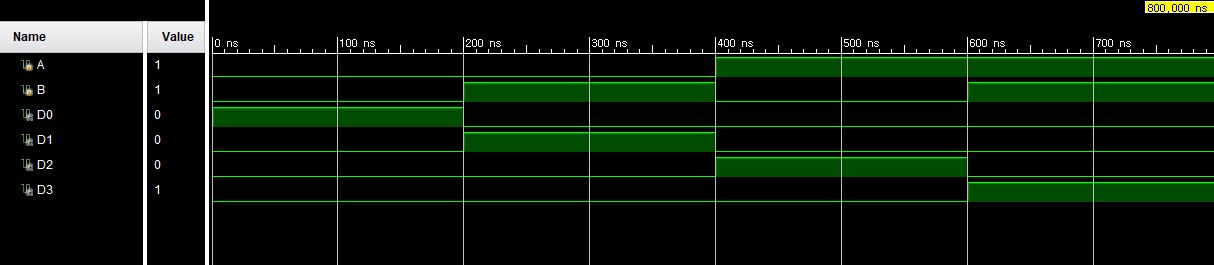
<Figure 1> AND 게이트를 이용한 2 to 4 decoder schematic diagram



<Figure 2> AND 게이트를 이용한 2 to 4 decoder simulation

위의 <Figure 1>은 1 bit 두 개의 입력, A와 B를 이용하여 AB라는 2-bit 이진수가 입력으로 들어왔을 때 AB가 00일 때는 D0의 출력이 1이 되고, AB가 01일 때는 D1의 출력이 1이 되고, AB가 10일 때는 D2의 출력이 1이 되고, AB가 11일 때는 D3의 출력이 1이 되는 2 to 4 decoder를 Verilog 코드로 구현하여 schematic을 확인한 결과입니다. 이때, D0은 A’B’, D1은 A’B, D2는 AB’, D3는 AB라는 논리 함수로 구할 수 있으므로 NOT 연산과 AND 연산만을 이용하여 Verilog 코딩하면 <Figure 1>과 같이 NOT 게이트 두 개와 AND 게이트 4개를 사용하여 2 to 4 decoder 논리 회로를 구현된다는 것을 알 수 있습니다. 또한, <Figure 2>를 보면 A, B에 대한 모든 입력에 대해 2 to 4 decoder의 동작을 예상했던 것과 동일한 simulation 결과를 확인할 수 있습니다.

<Figure 3> NAND 게이트를 이용한 2 to 4 decoder schematic diagram

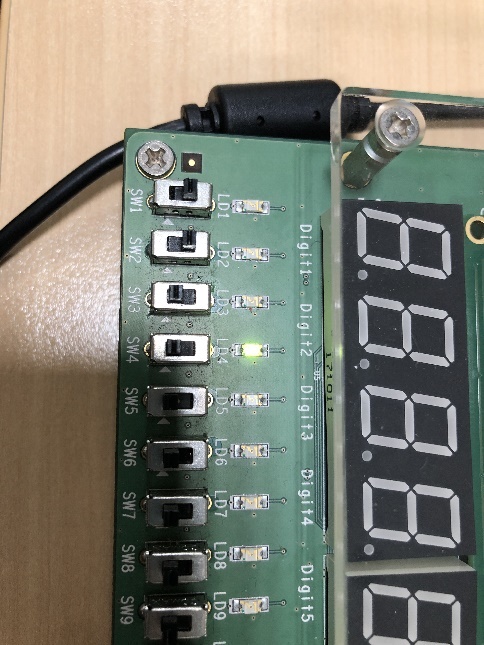
<Figure 4> NAND 게이트를 이용한 2 to 4 decoder simulation

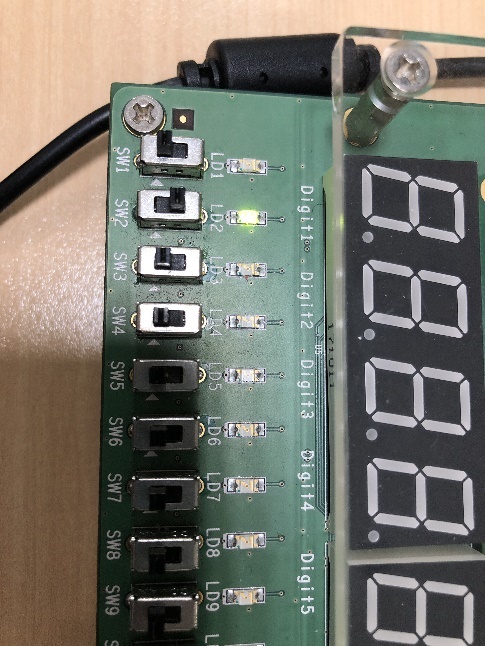
<Figure 3> 또한 <Figure 1>의 2 to 4 decoder 논리 회로를 AND 연산이 아닌 NAND 연산을 통해 구현한 회로의 schematic으로, NAND 연산은 NOT, OR, AND 연산을 모두 대체할 수 있으므로 NAND 연산을 이용해 2 to 4 decoder 논리 회로를 구현할 수 있었습니다. Verilog 코딩을 할 때 AND 연산과 NOT 연산을 이용해서 NAND 연산을 하였기 때문에 <Figure 3>과 같이 NAND 게이트 없이 AND 게이트와 NOT 게이트만으로 표현되었지만 AND 게이트와 그 바로 뒤에 오는 NOT 게이트가 합쳐져서 NAND 게이트를 의미한다고 보면 됩니다. <Figure3>의 Verilog 코드와 test bench를 이용하여 simulation을 확인해보면 <Figure 4>와 같고 이는 <Figure 2>와 같고, 구현하려고 했던 2 to 4 decoder의 동작과 동일하다는 것을 알 수 있습니다.

<Figure 2>와 <Figure 4>의 simulation 결과를 이용해 2 to 4 decoder의 진리표를 작성해보면 다음 <Table 1>과 같습니다.

<Table 1> 2 to 4 decoder 진리표

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | D0 | D1 | D2 | D3 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

A와 B 중 A가 MSB라고 가정하고 2 to 4 decoder 회로를 구현하였기 때문에 AB가 00일 때는 D0가 1이 되고, AB가 01일 때는 D1가 1이 되고, AB가 10일 때는 D2가 1이 되고, AB가 11일 때는 텍스트이(가) 표시된 사진

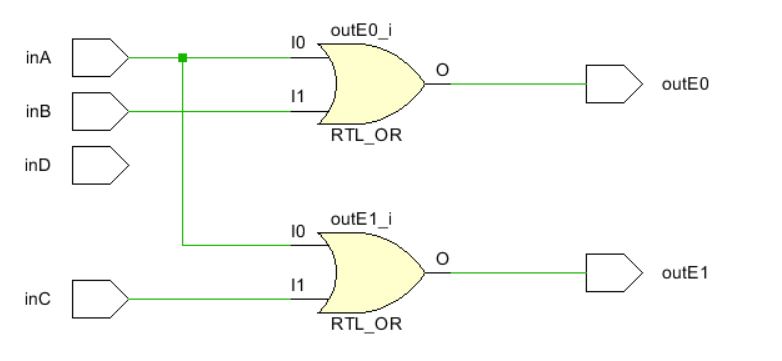
자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명D3가 1이 됩니다.

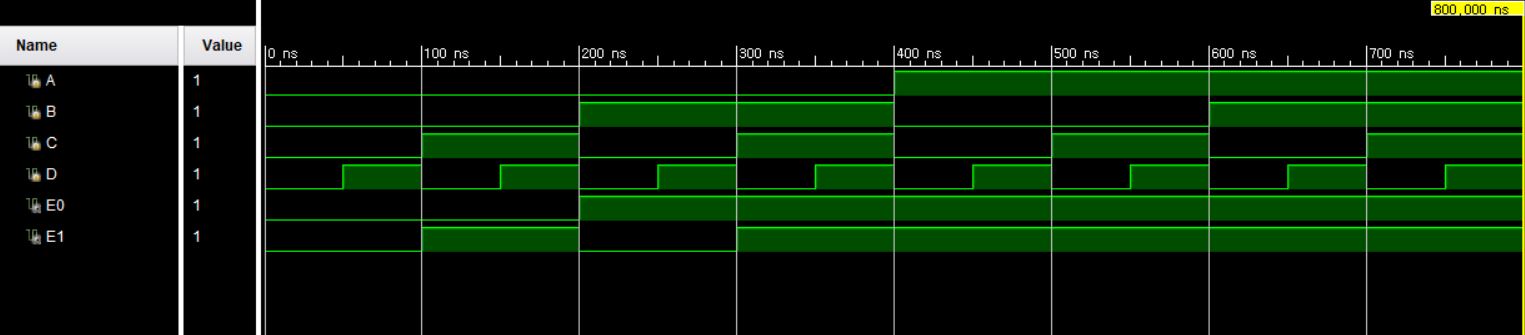
<Figure 5> FPGA를 통해 확인한 2 to 4 decoder의 동작

위의 <Figure 5>에서 SW1은 A를, SW2는 B를 의미하고, LD1은 D0, LD2는 D1, LD3는 D2, LD4는 D3를 의미합니다. 따라서, SW1와 SW2가 모두 0일 때는 LD1에 불이 들어오고, SW1와 SW2가 0, 1일 때는 LD2에, 1, 0일 때는 LD3에, 1, 1일 때는 LD4에 불이 들어오는 것을 확인할 수 있습니다.

1. 4 to 2 Encoder의 결과 및 simulation 과정에 대해서 설명하시오(Truth table 작성 및 k-map 포함).



<Figure 6> 4 to 2 encoder schematic diagram



<Figure 7> 4 to 2 encoder simulation

위의 <Figure 6>, 4 to 2 encoder schematic diagram은 A, B, C, D라는 네 개의 입력에 대해 E0, E1라는 두 개의 출력을 갖는 encoder를 Verilog 코드로 구현하여 schematic을 확인한 결과입니다. 이때, 입력 A, B, C, D는 각각 위에서 다룬 2 to 4 decoder의 출력 중 D3, D2, D1, D0과 같다고 같다고 가정하였고, 출력 E0와 E1는 E0를 MSB로 하는 2-bit 이진수라고 가정하고 코딩하였습니다. 따라서, ABCD가 0001일 때는 E0E1가 00이고, ABCD가 0010일 때는 E0E1가 01이고, ABCD가 0100일 때는 E0E1가 10이고, ABCD가 1000일 때는 E0E1가 11이 되도록 Verilog 코딩을 하였습니다. 따라서, E0은 A와 B의 논리 합으로, E1은 A와 C의 논리 합으로 표현할 수 있었습니다. 그런 다음, simulation 결과를 확인해보면 위의 <Figure 7>과 같았는데 A, B, C, D에 대한 모든 입력에 대한 simulation 결과를 확인하여 ABCD가 0001, 0010, 0100, 1000인 경우 외에도 다른 입력에 대해서도 어떠한 출력을 보이는지 확인할 수 있었습니다. <Figure 7>을 이용하여 4 to 2 encoder의 진리표를 작성해보면 다음 <Table 2>와 같습니다.

<Table 2> 4 to 2 encoder 진리표

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E0 | E1 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 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 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 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 |

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명<Table 2>에서 회색으로 표시한 부분이 일반적으로 encoder에 입력되는 입력에 대한 출력을 표시한 것입니다. 회색 부분을 보면 예측했던 대로 4 to 2 encoder가 동작한다는 것을 확인할 수 있습니다.

<Figure 8> FPGA를 통해 확인한 4 to 2 encoder의 동작

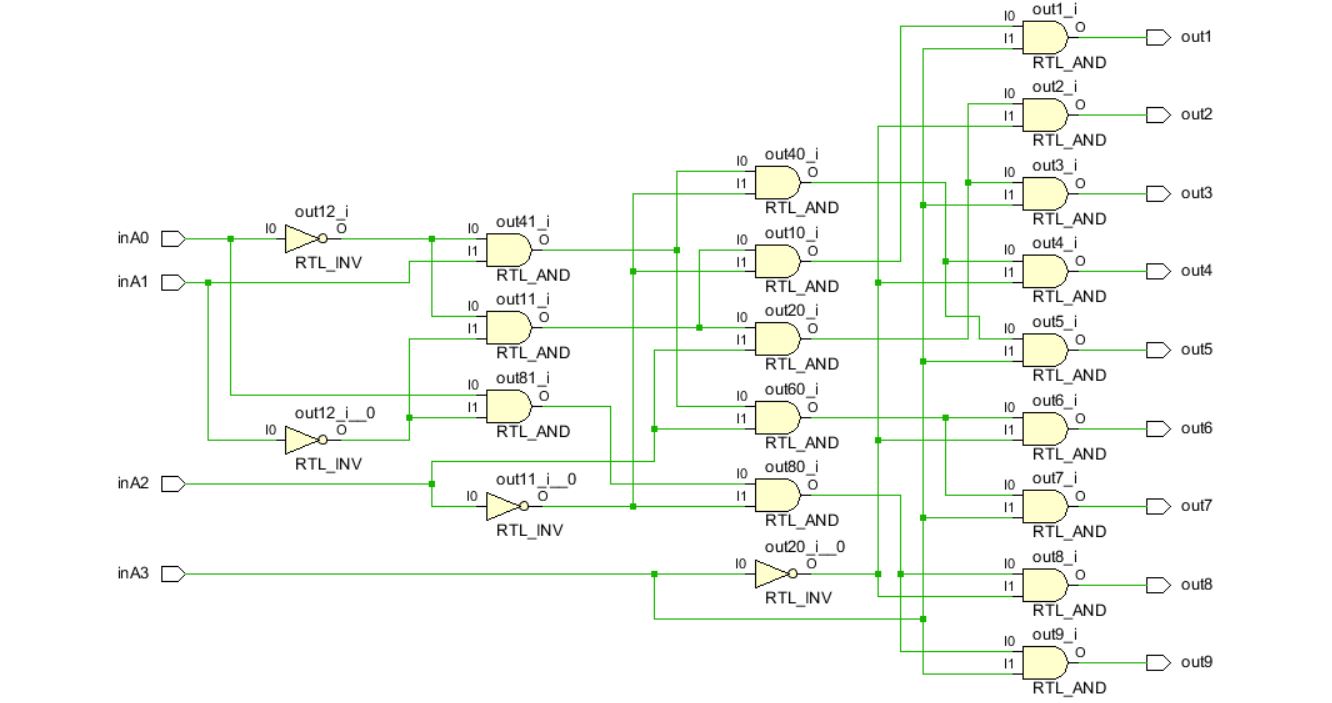
위의 <Figure 8>에서 SW1은 A를, SW2는 B를, SW3는 C를, SW4는 D를 의미하고, LD1은 E0, LD2는 E1을 의미합니다. 따라서, SW1가 1일 때는 LD1와 LD2에 모두 불이 들어오고, SW2가 1일 때는 LD1에만 불이 들어오고, SW3가 1일 때는 LD2에만 불이 들어오고, SW4가 1일 때는 모두 불이 들어오지 않는 것을 확인할 수 있습니다.

1. 4 to 2 Encoder에서 입력 형태 4가지를 제외한 나머지 입력 형태는 무엇을 뜻하는지 설명하시오.

위의 <Table 2>를 보면 4 to 2 encoder에 인가되는 일반적인 입력 4가지를 제외하고, 다른 입력들이 12가지가 존재한다는 것을 알 수 있습니다. 하지만 그러한 입력은 encoder가 예상하지 못한 입력이므로 인가되어서는 안 되지만 <Table 2>와 같이 입력이 된 경우에는 올바른 입력에 대한 출력과 구분할 수 없는 출력이 나올 수 있습니다. 예를 들어, encoder는 n개의 입력 중 반드시 하나의 입력이 1으로 인가되어야 하지만 ABCD가 0000으로 입력된 경우에는 E0=A+B=0+0=0으로, E1=A+C=0+0=0으로 출력되어 ABCD가 0001일 때와 동일한 출력을 보이게 됩니다.

1. 4 to 2 Encoder의 4가지 형태가 아닌 모든 입력 형태(16가지)에 대하여 동작되는 priority encoder의 논리 회로를 구성하시오.

위의 <Figure 6>가 4 to 2 encoder에 0001, 0010, 0100, 1000를 포함하여 모든 입력 형태에 대하여 동작되는 priority encoder라고 말할 수 있습니다. 이는 <Table 2>를 보면 회색 부분과 회색 부분 사이에 있는 입력들에 대한 출력을 확인하면 한 눈에 알 수 있는데, ABCD가 0100일 때와 1000일 때를 예시로 설명하면 ABCD가 0100일 때는 E0E1이 10이 되고 ABCD가 1000일 때는 E0E1이 11이 되는데 두 입력 사이에 있는 입력에 대해서는 E0E1이 10이라는 것을 알 수 있습니다. 이는 ABCD가 0101, 0110, 0111일 때 모두 E0E1이 10이 된다는 뜻으로, B, C, D 중 1인 입력이 여러 개 존재해도 B의 우선순위가 가장 높아 출력이 B에 의해서만 결정된다는 의미이기 때문입니다. 따라서, 네 개의 입력 A, B, C, D의 우선순위는 A, B, C, D 순이며 여러 개의 입력이 1일 때는 우선순위가 가장 높은 입력에 의해서만 출력이 결정됩니다.

1. BCD to Decimal decoder의 결과 및 simulation 과정에 대해서 설명하시오(Truth table 작성 및 k-map 포함).

<Figure 9> BCD to Decimal decoder schematic diagram

위의 <Figure 9>는 4-bit BCD를 의미하는 A0, A1, A2, A3라는 네 개의 입력을 이용하여 1부터 9까지의 Decimal로 바꿔주는 decoder를 구현한 verilog 코드의 schematic 결과입니다. 네 개의 입력 A0, A1, A2, A3은 16가지의 조합이 있지만 그 중 9개의 입력에 대해서 out1부터 out9 중 하나의 출력만이 1이 되는 decoder를 구현하였습니다.



<Figure 10> BCD to Decimal decoder simulation

<Figure 9>를 만든 Verilog 코드와 test bench를 이용해서 BCD to Decimal decoder의 simulation 결과를 확인해보면 위의 <Figure 10>과 같습니다. 이때, A0, A1, A2, A3이 0000이거나 1001보다 클 때는 out1부터 out9까지 중 어떠한 출력도 1이 되지 않는 것을 확인할 수 있고, A0A1A2A3이 0001일 때부터 1001일 때까지 out1부터 out9까지의 출력이 하나씩 1을 출력하는 모습을 확인할 수 있습니다. 이를 통해 진리표를 작성해보면 다음 <Table 3>와 같습니다.

<Table 3> BCD to Decimal decoder 진리표

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A0 | A1 | A2 | A3 | O1 | O2 | O3 | O4 | O5 | O6 | O7 | O8 | O9 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

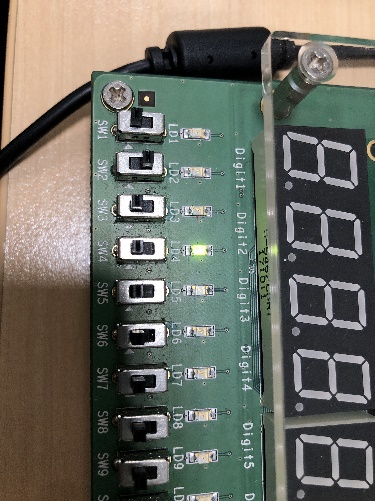
자동 생성된 설명위의 <Table 3>의 회색으로 표시해놓은 부분은 9개의 출력 중 하나의 출력이 1이 되는 경우를 표시한 것입니다. 강의자료에서는 네 개의 입력 중 A3가 MSB을 나타낸다고 가정하였지만, 4 to 2 encoder에서 숫자가 작은 출력(E0)을 MSB로 가정하였으므로 네 개의 입력 중 A0가 MSB을 나타낸다고 가정하였습니다.

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

<Figure 11> FPGA를 통해 확인한 BCD to Decimal decoder의 동작

위의 <Figure 11>에서 SW1은 A0를, SW2는 A1를, SW3는 A2를, SW4는 A3를 의미하고, LD1은 out1, LD2는 out2, LD3는 out3, LD4는 out4, LD5는 out5, LD6는 out6, LD7은 out7, LD8은 out8, LD9은 out9를 의미합니다. 따라서, 입력 BCD가 0001일 때는 LD1에, 0010일 때는 LD2에, 0011일 때는 LD3에, 0100일 때는 LD4에, 0101일 때는 LD5에, 0110일 때는 LD6에, 0111일 때는 LD7에, 1000일 때는 LD8에, 1001일 때는 LD9에 불이 들어오는 것을 확인할 수 있습니다.

1. Encoder와 decoder의 주요 응용에 대해서 설명하시오.

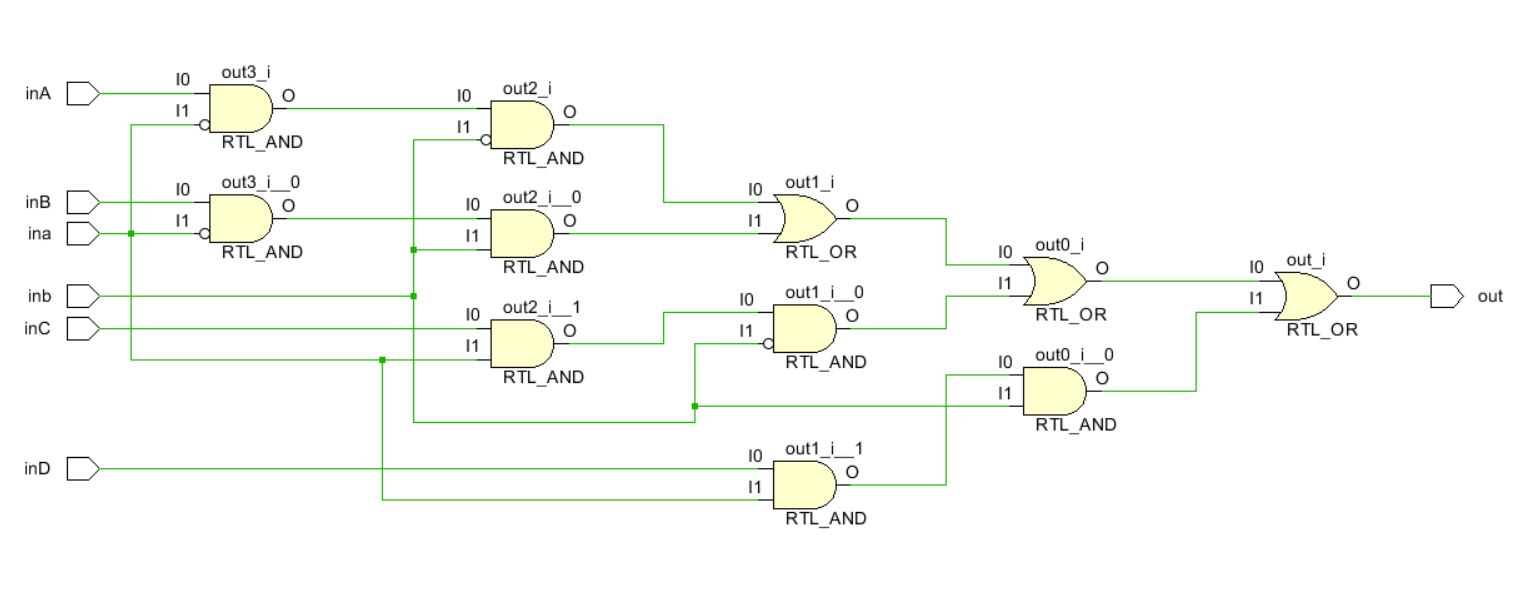
디코더는 각각의 주소를 가지고 있는 많은 장치들 중 하나를 고르는 역할을 하기도 합니다. 이때, 장치들의 각 주소는 디코더의 입력이 되고 하나의 출력이 active되면 해당 주소의 장치가 선택됩니다. 이러한 경우에는 디코더의 출력 중 단 하나만이 1이 되는 것이지만 8주차 실험에서 다뤘던 7-segment display을 동작시키는 회로의 경우, D, C, B, A의 4-bit 이진수 입력이 a, b, c, d, e, f, g의 7가지 출력이 되는 4 to 7 decoder를 구현한 것인데 D, C, B, A의 입력에 따라 a, b, c, d, e, f, g의 7개의 출력 중 하나만이 1이 되는 것이 아니라 여러 개의 출력이 1이 되는 decoder를 구현한 것입니다.

인코더는 디코더와 반대로 여러 개의 입력을 더 적은 수의 출력으로 만들어주는 논리 회로인데 Decimal to BCD encoder가 있다고 가정하면 I1부터 I9까지 들어온 9개의 입력에 대해 최소 4개의 출력으로 입력을 압축하여 반환할 수 있습니다.

1. 4 to 1 line MUX의 결과 및 simulation 과정에 대해서 설명하시오(code, truth table 작성).

텍스트이(가) 표시된 사진

자동 생성된 설명

<Figure 12> 4 to 1 line MUX Verilog code

<Figure 13> 4 to 1 line MUX schematic diagram

<Figure 14> 4 to 1 line MUX simulation

위의 <Figure 12>는 4 to 1 line MUX를 구현하기 위한 Verilog code로, 이를 이용하여 4 to 1 line MUX의 schematic diagram을 확인해보면 <Figure 13>과 같습니다. 4 to 1 line MUX는 select line을 의미하는 a, b, 두 개의 입력에 대해 a, b가 00이면 네 개의 입력 A, B, C, D 중 A에 의해서 출력이 결정되고 01이면 B에 의해서, 10이면 C에 의해서, 11이면 D에 의해서 출력이 결정되는 논리 회로를 의미합니다. 따라서, 출력 out은 a’b’A+a’bB+ab’C+abD라는 논리 함수로 표현할 수 있으므로 <Figure 12>와 같이 Verilog 코딩을 하였습니다. 4 to 1 line MUX의 simulation 결과인 <Figure 14>를 이용해 진리표를 작성해보면 다음 <Table 4>와 같습니다.

<Table 4> 4 to 1 line MUX 진리표

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a | b | A | B | C | D | Out |
| 0 | 0 | 1 | X | X | X | 1 |
| 0 | 0 | 0 | X | X | X | 0 |
| 0 | 1 | X | 1 | X | X | 1 |
| 0 | 1 | X | 0 | X | X | 0 |
| 1 | 0 | X | X | 1 | X | 1 |
| 1 | 0 | X | X | 0 | X | 0 |
| 1 | 1 | X | X | X | 1 | 1 |
| 1 | 1 | X | X | X | 0 | 0 |

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명a, b, A, B, C, D, 6개의 입력에 대해 가능한 입력의 가짓수가 64개나 되기 때문에 don’t care를 뜻하는 X를 이용하여 위의 <Table 4>와 같이 4 to 1 line MUX 진리표를 작성하였습니다. a, b는 select line이며 a가 MSB이고 A, B, C, D 중에서는 D가 MSD라고 가정하고 MUX를 구현하였습니다. 다중 입력 중 하나를 선택하여 하나의 출력선에 연결해주는 회로인 MUX는 ab의 값에 따라 네 개의 입력 중 한 개만을 통과시키는 스위치 같은 역할을 한다는 것을 알 수 있습니다. <Figure 14>의 simulation 결과를 보면 a, b가 00일 때는 입력 A와 출력 out이 동일하다는 것을 알 수 있고, 일 때는 입력 B와 출력 out이, 10일 때는 입력 C와 출력 out이, 11일 때는 입력 D와 출력 out이 동일하다는 것을 직관적으로 파악할 수 있습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

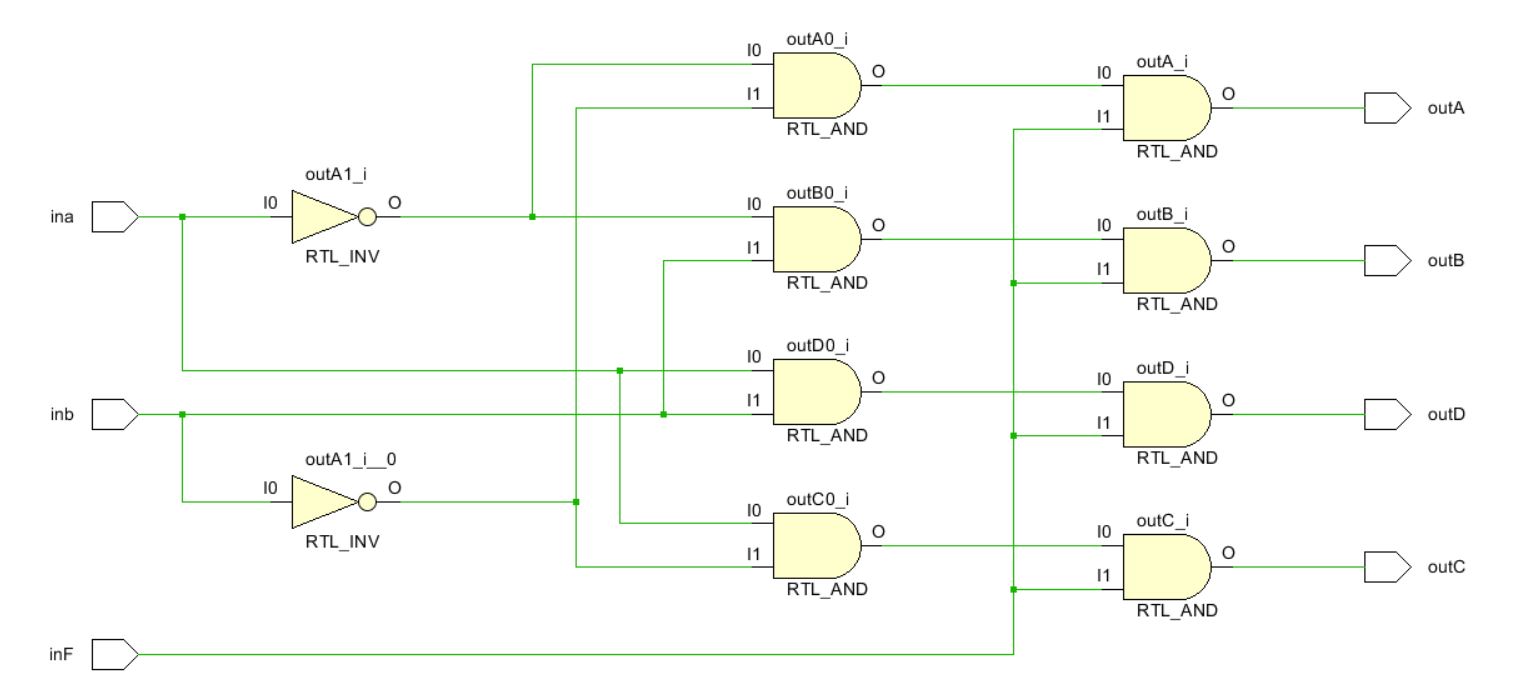
자동 생성된 설명<Figure 15> FPGA를 통해 확인한 4 to 1 line MUX의 동작

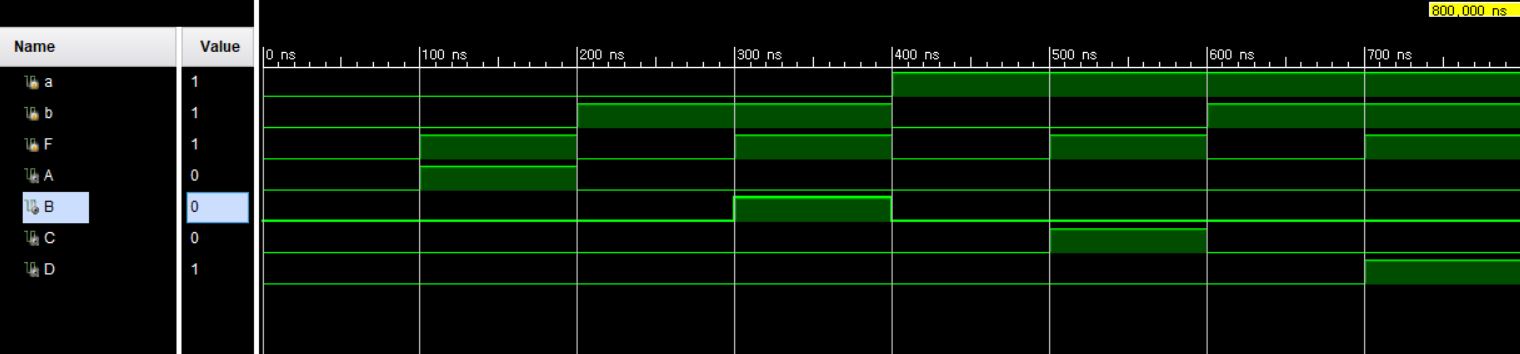
위의 <Figure 15>에서 SW1은 a를, SW2는 b를, SW3는 A를, SW4는 B를, SW5는 C를, SW6은 D를의미하고 LD1은 출력 out을 의미합니다. 따라서, ab가 00일 때는 A가 1일 때 LD1에 불이 들어오고, ab가 01일 때는 B가 1일 때 불이 들어오고, ab가 10일 때는 C가 1일 때 불이 들어오고, ab가 11일 때는 D가 1일 때 LD1에 불이 들어오는 것을 확인할 수 있습니다.

1. 1 to 4 line deMUX를 이용하여 4 to 16 decoder를 수행하고 결과를 나타내시오(code, truth table 작성).

텍스트이(가) 표시된 사진

자동 생성된 설명

<Figure 16> 1 to 4 line deMUX Verilog code

<Figure 17> 1 to 4 line deMUX schematic diagram

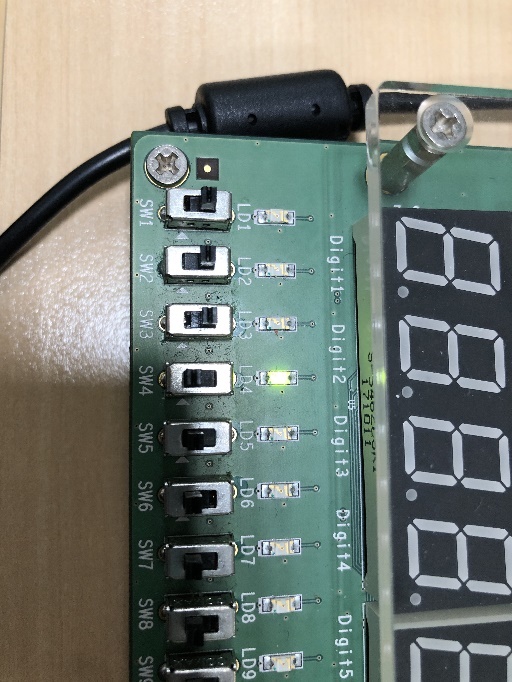
<Figure 18> 1 to 4 line deMUX simulation

위의 <Figure 16>은 1 to 4 line deMUX를 구현하기 위한 Verilog code로, 이를 이용하여 1 to 4 line deMUX의 schematic diagram을 확인해보면 <Figure 17>과 같습니다. 1 to 4 line deMUX는 select line을 의미하는 a, b, 두 개의 입력에 대해 a, b가 00이면 입력 F가 출력 중 A에 반영이 되고 01이면 입력 F가 출력 중 B에, 10이면 입력 F가 출력 중 C에, 11이면 입력 F가 출력 중 D에 반영되는 논리 회로를 의미합니다. 따라서, 출력 중 A는 a’b’F, B는 a’bF, C는 ab’F, D는 abF라는 논리 함수로 표현할 수 있으므로 <Figure 16>와 같이 Verilog 코딩을 하였습니다. 1 to 4 line deMUX의 simulation 결과인 <Figure 18>를 이용해 진리표를 작성해보면 다음 <Table 5>와 같습니다.

<Table 5> 1 to 4 line deMUX 진리표

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a | b | F | A | B | C | D |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |

위의 <Table 5>을 보면 ab가 00일 때는 출력 중 A가 입력 F와 동일하고, ab가 01일 때는 B가 입력 F와 동일하고, 10일 때는 C가 입력 F와 동일하고, ab가 11일 때는 D가 입력 F와 동일하다는 것을 확인할 수 있습니다. deMux에 경우에도 마찬가지로, a, b는 select line이며 a가 MSB이고 A, B, C, D 중에서는 D가 MSD라고 가정하고 deMUX를 구현하였습니다. Select line을 이용해 다중 출력 중 하나의 출력선을 연결하여 입력 F를 전달하는 deMux는 ab의 값에 따라 네 개의 출력 중 하나의 출력에 연결하여 입력을 전달하는, 입력을 분배하는 역할을 한다는 것을 알 수 있습니다.

텍스트, 전자기기, 검은색이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명<Figure 19> FPGA를 통해 확인한 1 to 4 line deMUX의 동작

위의 <Figure 19>에서 SW1은 a를, SW2는 b를, SW3는 F를 의미하고 LD1은 A, LD2는 B, LD3는 C, LD4는 D를 의미합니다. 따라서, ab가 00일 때는 F가 1일 때 LD1에 불이 들어오고, ab가 01일 때는 F가 1일 때 LD2에 불이 들어오고, ab가 10일 때는 F가 1일 때 LD3에 불이 들어오고, ab가 11일 때는 F가 1일 때 LD4에 불이 들어오는 것을 확인할 수 있습니다.

1. 결과 검토 및 논의 사항

2 to 4 decoder 실험을 통해 A, B 두 개의 입력에 대해 4가지 입력(00, 01, 10, 11)에 따라 D0, D1, D2, D3 중 하나의 출력선에서 1이 출력되는 것을 확인할 수 있었고, 반대로 4 to 2 encoder 실험을 통해서는 A, B, C, D 네 개의 입력이 decoder의 출력, D3, D2, D1, D0에 대응되므로 입력 중 하나만 1인 경우일 때 E0, E1이 00, 01, 10, 11 중 하나를 출력하는 것을 확인할 수 있었습니다. 이때, 4 to 2 encoder에서 네 개의 입력 중 여러 개의 입력이 1일 때는 A, B, C, D 순으로 priority를 가져 우선순위가 높은 입력에만 영향을 받아 출력이 결정되는 것 또한 확인할 수 있었습니다. BCD to Decimal decoder는 4-bit 입력이 1부터 9를 나타내는 값이면 9개의 출력 중 하나가 1로 결정되는 것을 확인할 수 있었는데 4-bit 입력은 16개의 값을 나타낼 수 있으므로 최대 16개까지의 출력선을 갖는 BCD to Decimal decoder를 구현할 수 있다는 것을 알 수 있었습니다. 4 to 1 line MUX는 2개의 select line이 존재할 때 4개의 입력선 중 하나만을 출력하는 스위치처럼 동작한다는 사실로부터 n개의 select line이 있는 MUX의 경우, 최대 2^n개의 입력선을 연결하여 그 중 하나를 출력할 수 있다는 것을 알 수 있었습니다. 1 to 4 line deMUX는 2개의 select line을 이용해서 F라는 입력을 4개의 출력선 중 하나로 전달하는 역할을 하였는데 이 또한 n개의 select line이 있을 때 최대 2^n개의 출력선을 연결하여 그 중 하나로 입력을 전달할 수 있다는 것을 알 수 있었습니다.

1. 추가 이론 조사 및 작성

4 to 2 encoder를 보면 A, B, C, D가 0000일 때와 0001일 때 모두 E0E1이 00을 출력하는 것을 확인할 수 있습니다. 따라서, 입력 중 참인 입력이 하나도 없어서 출력이 00인 것인지, 아니면 00을 의도적으로 출력한 것인지를 구분하기 위해서 추가적인 1-bit 출력을 사용할 수 있습니다. 4 to 2 encoder에서 사용한 A, B, C, D는 높은 자릿수를 D로 나타내는 것인지, A로 나타내는 것인지 딱 보고 알기가 힘들기 때문에 입력을 I3(MSB), I2, I1, I0이라고 하고 출력을 O1(MSB), O0라고 가정하고, 입력이 유효하다, 즉 입력 중 1인 입력이 1개 이상 있다는 것을 나타내기 위한 출력을 V라고 하면 4 to 2 priority encoder의 진리표는 다음과 같습니다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| I3 | I2 | I1 | I0 | O1 | O0 | V |
| 0 | 0 | 0 | 0 | X | X | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | X | 0 | 1 | 1 |
| 0 | 1 | X | X | 1 | 0 | 1 |
| 1 | X | X | X | 1 | 1 | 1 |

이때, 출력 V는 I3+I2+I1+I0라는 논리 함수로 표현할 수 있습니다.