9주차 예비보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20191599 이름: 송경호

**1.**

Decoder는 n bit의 이진 코드를 입력으로 받고, 이 입력에 대한 하나의 출력만을 활성화 (나머지 출력은 비활성화)하는 조합 회로이다. 출력은 주어진 입력에 따라 선택되며, 선택된 출력, 다시 말해 활성화된 출력은 다른 출력들과 다른값을 가진다. 이를 통해서 n bit를 입력을 받아 특정 2n bit로 변환하는 역할을 수행한다. 또한 Decoder은 Active Hight Decoder와 Active Low Decoder로 나눌 수 있다.

**1-1) Active High Decoder**

Active Hight Decoder은 특정한 하나의 출력이 high, 다시 말해 1의 값을 가지며 반대로 나머지 출력들은 모두 0의 값을 가지는 decoder을 칭한다. 예를 들어, 2 bit의 입력에 대해 4bit의 출력이 발생하는 Active High Decoder를 생각해보자. 이때 특정 입력값, 10에 대해 0100과 같이 출력의 특정비트만 1이 되고 나머지 비트는 0을 갖는다. 다음은 방금 설명한 decoder의 회로와 진리표이다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **입력** | | **출력** | | | |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

텍스트, 도표, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

진리표를 확인해보면 총 4개의 입력에 대해, 각 입력마다 출력 비트 중 한 비트만이 1의 값을 가지고 나머지 비트는 0을 가진다. 따라서 특정 입력에 대해서 특정 출력이 결정되어 이를 Decder라고 칭한다. 회로를 확인해보면 NOT과 AND 회로를 통해 Acitve High Decoder를 구현했다.

**1-2) Active Low Decoder**

Active Low Decoder은 위와 반대로 하나의 입력에 대해 출력 중 오직 하나의 비트만이 Low인 값, 다시 말해 0의 값을 가지며 나머지 출력 비트는 모두 1을 갖는다. 아래는 진리표와 회로도이다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **입력** | | **출력** | | | |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

텍스트, 도표, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

진리표를 봤을 특정 입력에 대해서 하나의 출력 비트만이 0의 값을 갖고 나머지 출력 비트는 모두 1의 값을 가지는 것을 알 수 있다. 또한 회로도의 구성 역시 Active High Decoder와 기본적으로 동일하지만 AND가 아닌 NAND 게이트를 통해 결과값을 부정함으로써 반대의 결과를 얻게 된다. 이때 NAND gate가 AND gate에 비해 적은 수의 트랜지스터를 사용하기 때문에 여러 측면에서 Low Decoder가 High Decoder에 비해 효율적일 수 있다.

추가적으로 Decoder 중 enable 제어신호 EN을 입력으로 받는 경우가 있다. 이때 enable 제어신호는 회로의 작동을 허가할지 혹은 허가하지 않을지 결정하는 역할을 한다. 따라서 Enable 비트가 high일 경우 회로는 작동하지 않는다. 회로도는 다음과 같다.

도표, 라인, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

Decoder은 메모리의 주소를 입력받아 특정 데이터를 선택하고 해당 데이터를 읽거나 쓸 수 있는 신호를 생성하는 메모리 주소 디코더에 많이 사용된다.

**2.**

Encoder은 이름에서 알 수 있듯이, Decoder의 반대과정을 수행한다. 다시 말해 2n bit의 입력을 n bit의 출력으로 변환하는 역할을 수행한다. 다음은 encoder의 진리표 및 회로 예시이다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **입력** | | | | **출력** | |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |

도표, 폰트, 라인, 스케치이(가) 표시된 사진

자동 생성된 설명

예를 들어 0001이라는 22bit 입력이 들어왔다고 가정했을 때 Encoder은 이를 00이라는 2 bit 출력으로 변환하는 것을 알 수 있다. 또한 회로에서 주목해야 할 점은 D0의 입력값이 완전히 무시된다는 점이다. Encode는 암호화라는 의미를 함유하고 있지만, 단순히 암호화를 넘어 데이터를 변환하고, 표준화하며, 압축하는 등 다양한 분야에서 사용할 수 있다.

**3.**

Mux는 multiplexer으 준말이며, 여러 입력 중하나를 선택하여 출력으로 전송하는 디지털 논리 회로이다. Mux의 특징은 다수의 입력에 대해 하나의 출력만을 갖는다는 점이다. 또한 Mux는 선택할 후보가 되는 입력 외에 select input이라는 입력이 추가로 존재하는데, 이는 여러 입력 중 어떤 입력을 택하여 출력에 전달할지 결장하는 역할을 한다. 이때 입력이 2nbit라면 select input은 n bit가 된다. 예를 들어 4x1 mux의 경우 4개의 입력을 바탕으로 하나의 출력을 결정하며 이때 select input는 2 bit가 된다. 입력 D0 ~ D3에 대한 4X1 mux의 진리표와 회로도이다,

|  |  |  |
| --- | --- | --- |
| **S1** | **S0** | **Output** |
| 0 | 0 | D0 |
| 0 | 1 | D1 |
| 1 | 0 | D2 |
| 1 | 1 | D3 |

텍스트, 도표, 스크린샷, 그래프이(가) 표시된 사진

자동 생성된 설명

진리표를 확인하면 select input bit의 조합을 통해 출력이 결정되는 것을 확인할 수 있다. 또한 추가적으로 4x1 mux의 경우 2x1 mux 3개의 조합으로 나타낼 수 있다.

도표, 기술 도면, 평면도, 개략도이(가) 표시된 사진

자동 생성된 설명

회로도에서 총 2개의 mux를 통해 4개의 입력을 받는다. 해당 mux들은 2x1 mux이므로 각 각 한개의 입력을 출력으로 전달하고 마지막 2x1 mux를 통해 둘 중 하나를 출력으로 전달하여 결과적으로 4x1 mux의 역할을 수행할 수 있게 된다. 같은 방법으로 8x1을 4x1 mux로 구현할 수 있고 이들은 역시 2x1 mux로 재귀적으로 구현이 가능하다.

**4.**

Priority Encoder은 우선순위 encoder이다. 해당 회로는 여러 개의 입력 중 가장 높은 우선 순위를 가진 입력 신호를 식별하여 해당 입력 신호에 대한 출력을 생성한다. 다시 말해 여러 입력이 활성화되었을 때, 우선 순위에 따라 하나의 입력만을 선택할 수 있도록 한다.

일반적인 encoder의 경우에는 여러 개의 입력 신호에 한번 에 1이 들어온 경우에 대한 처리를 할 수 없지만 priority encoder는 우선순위를 통해 이를 가능하게 한다. 또한 일반적인 encoder의 경우 입력 신호가 0001이어서 출력이 00인 경우와, 입력 신호가 존재하지 않아 출력이 00인 경우에 대한 처리가 불가능한데, priority encoder의 경우 적어도 하나의 비트가 high인 경우 high가 되는 추가적인 flag 입력 bit를 통해 이에 대한 처리가 가능해진다. 다음은 회로도와 진리표이다.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **입력** | | | | | **출력** | | | |
| **Ei** | **D3** | **D2** | **D1** | **D0** | **a** | **b** | **G** | **Eo** |
| 0 | X | X | X | X | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | X | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | X | X | 1 | 0 | 1 | 0 |
| 1 | 1 | X | X | X | 1 | 1 | 1 | 0 |

**도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명**

해당 priority encoder 에시는 priority가 D3, D2, D1, D0 순으로 높은 경우이다. X의 경우 don’t care bit를 나타내며 만약 D3 bit이 1인 경우, D3의 우선순위가 가장 높기 때문에, 나머지 비트는 high와 low 여부가 결과값에 아무런 영향을 미칠 수 없게 된다. 추가적으로 Ei 비트가 앞서 언급한 flag 역할을 하며 Ei비트가 0인 경우 모든 입력 비트가 Don’t care bit가 되는 것을 알 수 있다.

**5.**

Demux는 Demultiplexer로, encoder와 decoder의 관계처럼 mux의 반대되는 역할을 한다. 다시말해 하나의 입력 신호를 받아 여러 개의 출력을 만들어낸다. DeMux와 Decoder는 하나의 입력에 대해 여러 출력을 만들어낸다는 공통점이 있지만 Select input의 유무에 대해 차이가 있다.

Demux의 경우 mux와 마찬가지로 select input을 바탕으로, 출력을 선택한다. 따라서 동일한 입력이라도 Select input에 따라 그 출력값이 바뀐다.

반면에 Decoder는 Select input의 개념이 존재하지 않는다. 따라서 동일한 입력에 대해 반드시 동일한 출력이 발생하게 된다.

도표, 라인, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명

Decoder는 입력을 다른 형식으로 변환하여 수신자의 입력에 대한 이해를 돕는 역할을 하기 때문에 select input의 차이점을 바탕으로 용도의 차이가 발생하게 된다.

**6.**

**6-1) Decoder 활용 예시 - ROM**

앞서 말했듯이 decoder는 메모리 접근에 다양하게 사용된다. ROM(Read Only Memory)은 내부적으로 decoder를 사용한다. User가 메모리의 특정 주소에 저장된 데이터를 읽기 위해 n bit 의 주소 체계 신호를 컴퓨터에 보내면, 이 주소를 기반으로 ROM은 해당 주소에 저장된 데이터 중 하나를 선택하여 출력한다.

**6-2) Encoder 활용 예시 - 파일 압축**

컴퓨터에서 데이터의 양을 줄이기 위해 데이터를 코드화하고 압축하는데 encoding이 활용된다.

**6-3) Multiplexer 활용 예시 – 라디오 튜너**

라디오는 여러 방송국의 신호 중에서 하나를 선택한다. 라디오의 튜너를 돌려 방송을 맞추는 행위는 입력을 선택하는 Select input을 지정하는 것이다. 즉 여러 개의 입력 중에서 Select input으로 특정 입력 신호를 선택하고, 그 출력을 스피커로 송출하게 된다.

**6-4) Demultiplexer 활용 예시 – 통신 시스템**

통신 시스템은 신호의 송신 및 수신을 위해 송신은 mux, ​​수신은 demux를 이용한다. 신호의 전송은 mux 출력으로 수행되고, 이는 다시 demux의 입력으로 제공되며 수신기는 이 입력을 다시 원래 신호로 변환된다.

인코딩과 디코딩에 대한 추가적인 사례도 존재한다.

|  |  |  |
| --- | --- | --- |
|  | **인코딩** | **디코딩** |
| 통신보안 | 암호화 | 암호 해제로 원래의 신호로 변환 |
| 아파트 층을 숫자화 | 십진수화 | 해당 층을 지목 |
| 음성 신호 처리 | 전기신호로 변환 | 물리적 신호로 변환 |
| 디지털화 | 이진화 | 아날로그로 변환 |
| 통신 라인의 특성에 맞춤 | 라인코딩 | 원래 직렬데이터로 변환 |