# 컴퓨터 공학 기초 실험2 보고서

## 실험제목: 2-to-1 MUX

## 실험일자: 2023년 09월 11일 (월)

## 제출일자: 2023년 09월 20일 (수)


학 과: 컴퓨터공학과

담당교수: 이준환 교수님

실습분반: 월요일 0, 1, 2

학 번: 20222020264

성 명: 최봉규

# 1. 제목 및 목적

## A. 제목

2-to-1 MUX

## B. 목적

Quartus의 기본 사용법을 익히고, Verilog를 이용해 2-to-1 MUX를 디자인한다. Boolean equation을 이용하여 mux의 sub module 로 2 input NAND gate와 Inverter를 디자인 후 조합하여 2-to-1 MUX를 구현한다. 구현 이후 testbench를 통해 본인이 구현한 2-to-1 MUX가 올바르게 작동하는지 확인한다.

# 2. 원리(배경지식)

Verilog HDL은 Verilog Hardware Description Language로 하드웨어를 이와 같은 언어로 설계하고 구현, 검증이 가능하다. 사용자 입장에서 회로 설계, 관리가 용이하다. Verilog system에서의 값은 0, 1, z, x 로 4개의 값을 가질 수 있다. 0은 Logical low, ground, VSS를 의미한다. 표기상 0이지만 이는 퍼텐셜 에너지로 기준점이 되는 것일 뿐이다. 1은 Logical high, power, VDD를 의미한다. 0을 기준점으로 상대적으로 전기 에너지가 크다는 사실을 알 수 있다. Z는 High-impedance, float을 의미한다. 저항이 너무 커 알 수 없다. 0인지 1인지 정의하기 어렵다. 또는 도선이 끊어져 값을 알 수 없는 경우에 z값이 된다. X는 unknown, don't care이나 주로 don't care의 의미로 사용하곤 한다. z처럼은 아니지만 어떤 값인지는 모르지만 상관없을 경우에 주로 x로 표기 가능하다.

Verilog 또한 언어이다 보니 C/C++처럼 변수 이름 설정이 가능하다. 앞에선 변수라고 설명했지만 모듈, 포트, 변수 등을 설정이 가능하다. 이름은 알파벳이나 언더스코어('_')로 먼저 시작이 가능하다. 이름은 1023자까지 가능하다. 처음만 아니라면 숫자와 달러 표시($) 도 넣을 수 있다. 언어 특성상 대소문자를 구분하여 ad2와 AD2를 구분하여 다른 이름으로 구별한다. 주석 기입도 가능하다. 주석은 C/C++ 처럼 // 표시를 이용해서 한 줄 주석이 가능하고 복수의 줄은 /* */ 표시로 주석 표시가 가능하다.

Verilog의 module 은 C++의 class처럼 생각할 수 있다. 객체의 이름이 있고 파라미터 값을 주면 call by reference 같은 느낌으로 값을 module 내에서 수정할 수 있다. Module를 정의할 때는 module "module_name" ("input_output_port_list")의 형식으로 진행한 다음 파라미터를 input과 output으로 설정하고 함수를 만들듯이 만든다. 이때 assign이나 = 같은 형식을 이용하여 만들 수 있다. 모든 작성이 끝난 후에는 endmodule로 module 정의를 끝내면 module 하나를 만드는 것이다.

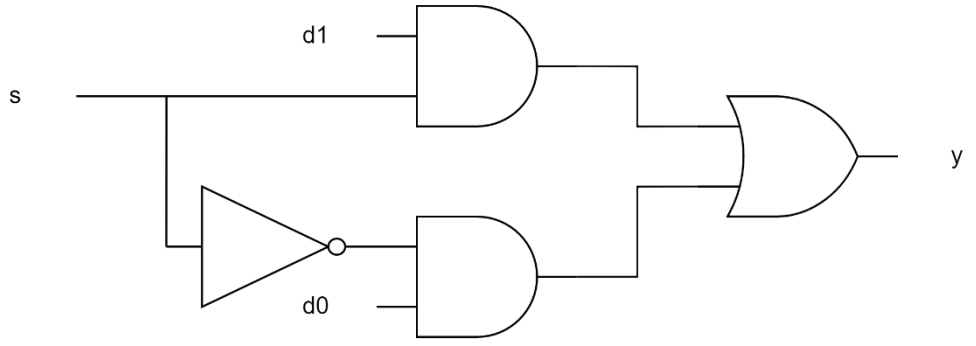
MUX는 select input을 이용하여 여러 input data 중 하나를 출력으로 내보내는 역할을 수행한다. Input의 data 수에 따라 bit를 조절 가능하다. 2^n 개의 input data들 중 하나를 선택하는 select input은 최소 n bit로 구성되어야 MUX의 역할을 수행할 수 있다. 해당 과제는 2 to 1 MUX로 최소 1bit의 bit수가 필요하다. 이번 과제는 2 inputs nand gate와 inverter를 구현하여 적절한 조합으로 MUX를 구현한다. 2 intputs nand gate는 두 개의 입력값을 받아 AND 연산을 한 후 거기에 NOT을 붙인 값으로 출력값을 갖게 된다. 이는 부울 식으로 input a, b 와 output y 가 있다고 가정한 후 y = !(a & b)의 식을 가지게 된다. Inverter에 경우에는 하나의 입력값을 받은 후 출력값으로 입력값에 NOT을 붙여준 값을 주게 된다. Input a 와 ouput y 가 있다고 가정했을 때 y = ! a의 식이 된다.

## 3. 설계 세부사항

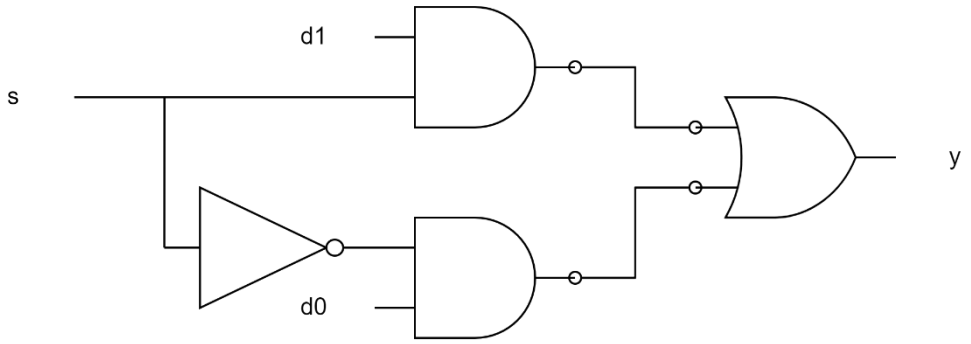
이번 Verilog로 설계한 2 to 1 MUX는 input port로 d0, d1, s를 가지고, output port로 y를 갖는다. 이 MUX는 다음과 같은 진리표를 가진다.

| d1d0 ＼ S | 0 | 1 |
|---|---|---|
| 00 | 0 | 0 |
| 01 | 1 | 0 |
| 11 | 1 | 1 |
| 10 | 0 | 1 |

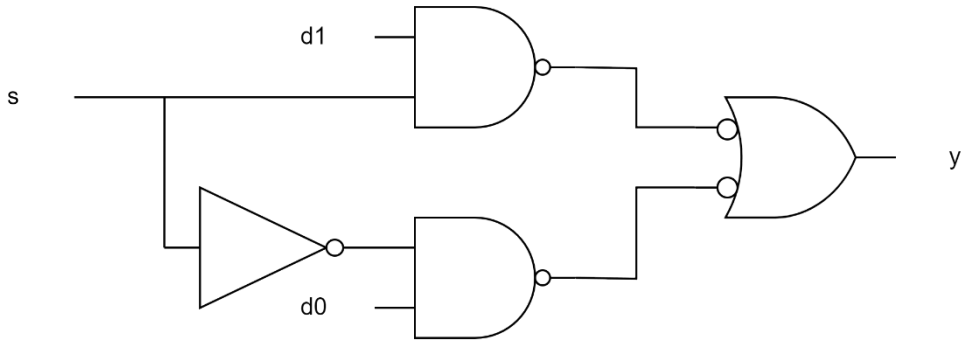
K map을 이용한 결과 output y = (d0&~s) | (d1$s) 임을 확인할 수 있다. 이 식을 게이트를 이용해서 표현하게 되면 아래의 그림과 같다.



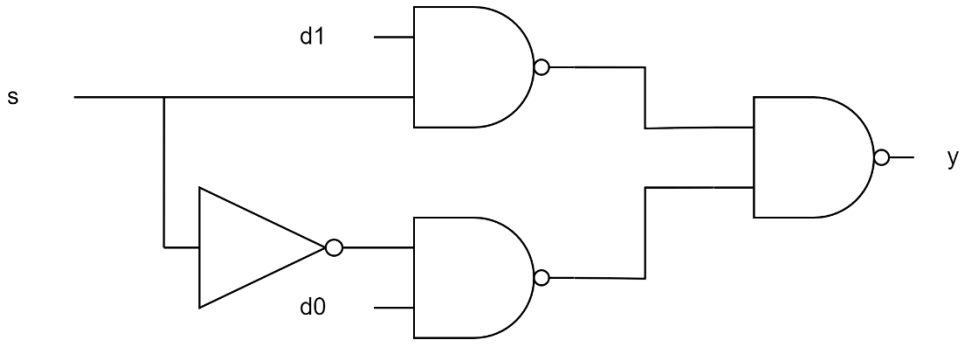

AND gate 와 OR gate는 같은 수의 input NAND gate와 OR gate 대비 트랜지스터(n mos, p mos)의 개수가 두 개 더 필요하게 도니다. 2 input gate의 경우 앞 bubble이 없는 경우의 트랜지스터 개수가 있는 것의 1.5 배이다. 코스트를 줄이기 위해 드 모르간 법칙을 이용할 수 있다. 다음은 위의 그림을 bubble pushing을 하며 최적의 MUX 구현을 위한 트랜지스터 개수를 줄이는 단계이다.




먼저, 중간 wire 부분에 양쪽에 bubble을 주면서 회로의 Boolean equation이 바뀌지 않도록 한다. 좀 더 보기 쉽게 표현하면 아래의 그림과 같다.

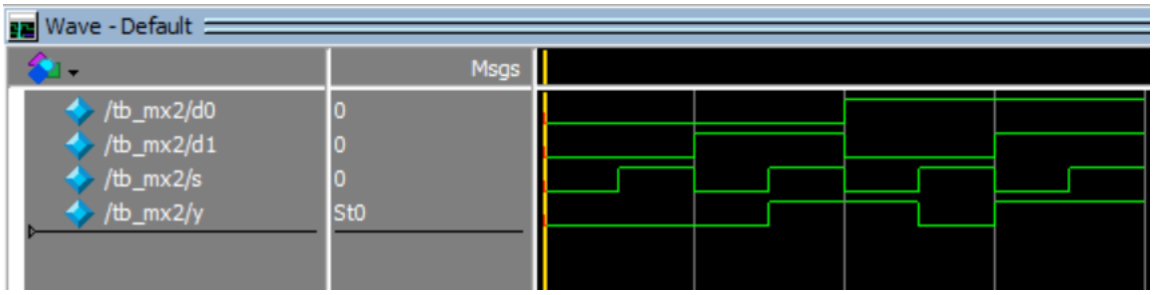



다음은 bubble pushing 하는 과정이다. 우측에 위치한 OR gate를 bubble pushing을 하게 되면 input에 있던 bubble이 output 쪽에 위치하게 된다. 또한 OR gate 가 AND gate로 바뀌게 된다. 최종적으로 두 개의 bubble이 붙은 input을 OR gate에 연결한 것은 NAND gaet의 모습이 됨을 아래 그림으로 확인 가능하다.



## 4. 설계 검증 및 실험 결과

### A. 시뮬레이션 결과



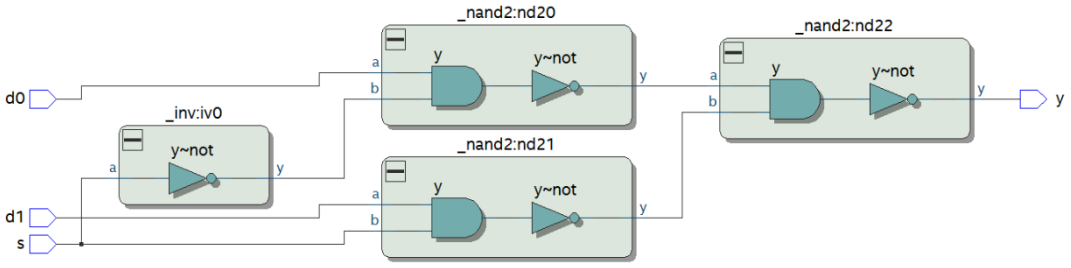

웨이브 폼의 입력값은 아래의 표의 순서와 동일하게 넣었다.

| ID | Input condition | Expected value | Measured value | comparison |
|---|---|---|---|---|
| 1 | d0 = 1'b0<br>d1 = 1'b0<br>s = 1'b0 | y = 1'b0 | y = 1'b0 | corrected |
| 2 | d0 = 1'b0<br>d1 = 1'b0<br>s = 1'b1 | y = 1'b0 | y = 1'b0 | corrected |
| 3 | d0 = 1'b0<br>d1 = 1'b1<br>s = 1'b0 | y = 1'b0 | y = 1'b0 | corrected |
| 4 | d0 = 1'b0<br>d1 = 1'b1<br>s = 1'b1 | y = 1'b1 | y = 1'b1 | corrected |
| 5 | d0 = 1'b1<br>d1 = 1'b0<br>s = 1'b0 | y = 1'b1 | y = 1'b1 | corrected |
| 6 | d0 = 1'b1<br>d1 = 1'b0<br>s = 1'b1 | y = 1'b0 | y = 1'b0 | corrected |
| 7 | d0 = 1'b1<br>d1 = 1'b1<br>s = 1'b0 | y = 1'b1 | y = 1'b1 | corrected |
| 8 | d0 = 1'b1<br>d1 = 1'b1<br>s = 1'b1 | y = 1'b1 | y = 1'b1 | corrected |

모든 경우의 수에 대하여 input 값이 3개가 있으니 최대 8개의 경우의 수가 나오는 것

을 확인할 수 있다. 이에 대해 모든 경우의 수를 테스트 벤치를 통해 확인한 결과 예상 값과 실측값 모두가 일치하는 것을 확인할 수 있었다.

## B. 합성(synthesis) 결과




RTL map viewer를 통해 합성된 결과를 보자면 첫번째 NAND gate인 nd20에서 d0 와 !s인 sb가 연결된 것을 두번째 NAND gate인 nd21에서 d1과 s가 연결된 것을 확인할 수 있다. 마지막으로 세번째 NAND gate인 nd22에서 이전에 사용했던 두 NAND gate의 결과가 입력값으로 들어가 y가 출력됨을 확인할 수 있다.

**Flow Summary**

<<Filter>>

| | |
|---|---|
| Flow Status | Successful - Fri Sep 08 20:49:18 2023 |
| Quartus Prime Version | 18.1.0 Build 625 09/12/2018 SJ Lite Edition |
| Revision Name | mx2 |
| Top-level Entity Name | mx2 |
| Family | Cyclone V |
| Device | 5CSXFC6D6F31C6 |
| Timing Models | Final |
| Logic utilization (in ALMs) | 1 / 41,910 ( < 1 % ) |
| Total registers | 0 |
| Total pins | 4 / 499 ( < 1 % ) |
| Total virtual pins | 0 |
| Total block memory bits | 0 / 5,662,720 ( 0 % ) |
| Total DSP Blocks | 0 / 112 ( 0 % ) |
| Total HSSI RX PCSs | 0 / 9 ( 0 % ) |
| Total HSSI PMA RX Deserializers | 0 / 9 ( 0 % ) |
| Total HSSI TX PCSs | 0 / 9 ( 0 % ) |
| Total HSSI PMA TX Serializers | 0 / 9 ( 0 % ) |
| Total PLLs | 0 / 15 ( 0 % ) |
| Total DLLs | 0 / 4 ( 0 % ) |

Flow Status 항목에서 합성이 잘 돼있음을 알 수 있다.

# 5. 고찰 및 결론

### A. 고찰

이번 실습에서는 2 to 1 MUX를 구현했다. 구현함에 있어서 특별히 신경 쓸만한 문제는 발생하지 않았지만, Quartus를 오랜만에 돌려 탑 모듈 설정이나 testbench 설정에서 헤맸다. gate를 구성하고 MUX를 구현했을 때 잘 안되어 top module 설정이 이상하게 되어있었고, behavior gate로 이미 같은 이름의 모듈이 있어 오류가 난 것으로 판단했다. 이 문제를 내가 설정한 모듈 앞에 언더스코어를 붙이며 이와 같은 문제를 해결했다. top모듈까지 settings에서 맞게 설정하고 나서 RTL map viewer를 통해 제대로 합성되었는지 확인했다. 후에 testbench를 작성할 때에는 크게 막힘 없이 했으며 wave form을 통해 wire와 reg가 제대로 설정된 것을 확인할 수 있었다.

### B. 결론

nand gate나 inverter를 굳이 구현하지 않고 식을 바로 assign을 해도 괜찮을 것 같으나 굳이 따로 gate를 먼저 구현하고 mux를 구현하는 것은 효율이 좀 떨어진다고 생각한다. 그럼에도 이러한 방법은 코드의 가독성을 높이기 위한 장치 역할인가를 생각해 보았다. 그리고 이런 방법을 쓰면서 파일 분할 같은 방법을 터득할 수 있어 이번 실습의 내용이 괜찮았다고 생각한다.

## 6. 참고문헌

이준환 교수 / 디지털논리회로2 / 광운대학교(컴퓨터정보공학부) / 2023

도움을 준 사람(들):

- 검증표 제작을 위해 2022202067 김태관의 도움을 받음.