컴퓨터 공학 기초 실험2 보고서

실험제목: 2-to-1 MUX

실험일자: 2023년 09월 11일 (월)

제출일자: 2023년 09월 15일 (금)

학 과: 컴퓨터정보공학부

담당교수: 이준환 교수님

실습분반: 월요일 0, 1, 2

학 번: 2022202065

성 명: 박나림

1. 제목 및 목적

A. 제목

2-to-1 MUX

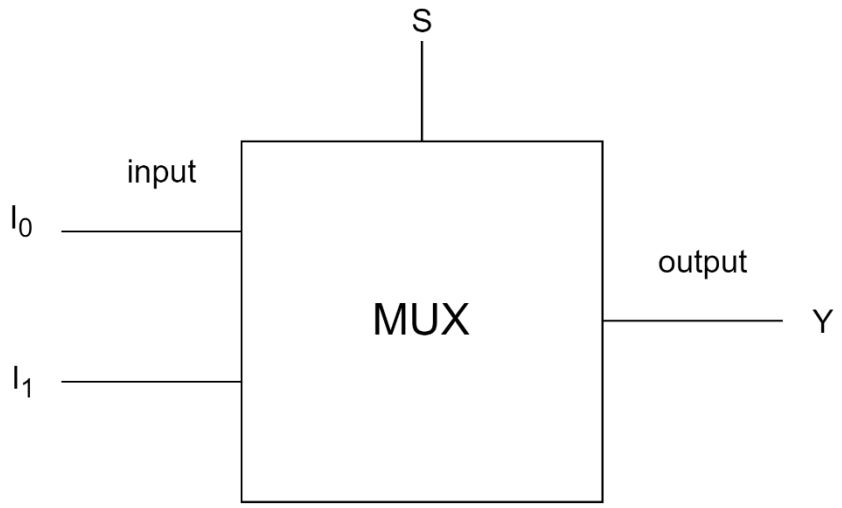
B. 목적

NAND gate와 inverter gate를 이용하여 MUX(Multiplexer)를 설계한다. 베릴로그 문법을 공부하여 MUX 코드를 짤 수 있도록 한다. testbench를 통해 설계한 결과를 검증하여 확인까지 마치도록 한다.

2. 원리(배경지식)

A. MUX

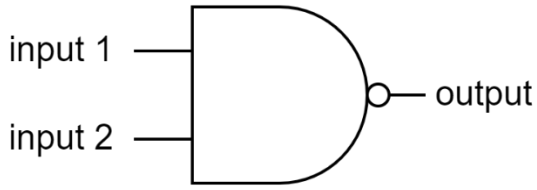
MUX(Multiplexer)는 여러 개의 아날로그 또는 디지털 입력 신호들을 input으로 받아 그 중 하나를 선택하여 output으로 출력하는 조합 회로 장치이다. 아래는 2 to 1 MUX의 예시이다.



이러한 멀티플렉서를 사용하면 여러 연결들을 한 채널로 담을 수 있기 때문에 비용을 절감할 수 있다는 장점이 있다. DeMUX, 디멀티플렉서는 이와 반대로 하나의 입력들을 여러 개의 출력으로 내보내는 역할을 한다.

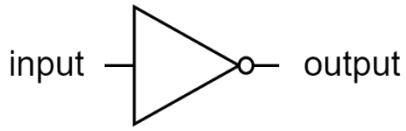
B. NAND gate & Inverter gate

-NAND gate



AND gate의 반대의 결과를 내는 게이트로, 모든 입력들이 참일 때에만 결과로 거짓을 출력하는 논리 회로이다. 베릴로그 문법으로 표현하면 output = ~ (input1 ^ input2); 로 나타낼 수 있다. input들의 값을 AND 연산(^)하고, 그 전체에 NOT 연산(~)을 한 것이다.

-Inverter gate



NOT 게이트로 부르기도 하며, 입력의 반대의 값이 출력되는 논리 회로이다. 베릴로그 문법으로는 NOT 연산인 (~)으로 표현하여, output = ~input; 으로 사용할 수 있다.

C. Verilog HDL

베릴로그에서는 기본적으로 우선 top module을 생성하여 프로젝트를 시작한다. 이때 top module의 이름과 컴파일 되는 파일의 module 이름은 서로 일치해야 한다. 또한 module 이름을 쓰는 부분에서도 ';'을 붙여야 하며, module module_name (변수들); 순서로 쓰도록 한다. 이때 변수들의 순서에 맞춰서 다른 데에서 module을 사용할 때 같은 순서로 변수들을 써야 한다. 이때에는 정의된 module_name 새로운 module_name (.정의된 변수(새로운 변수)); 식으로 사용한다. 순서만 맞으면 정의된 변수를 쓰는 과정은 생략해도 된다.

모든 module은 endmodule로 끝나며 한 줄을 띄어야 한다. 또한 begin, end에서 쓰이는 end와 endmodule에서는 ';'을 붙이지 않도록 한다.

testbench까지 만들면 시뮬레이션을 돌려 waveform을 볼 수 있으며, 자신의 코드를 검증할 수 있다.

3. 설계 세부사항

NAND gate 3개, Inverter gate 1개를 이용하여 MUX를 설계한다.

-NAND gate (module _nand2)

1) 입출력

I / O	변수명
Input	a
	b
Output	y

2) 진리표

a	b	y
0	0	1
0	1	1
1	0	1
1	1	0

-Inverter gate (module _inv)

1) 입출력

I / O	변수명
Input	a
Output	y

2) 진리표

a	y
0	1
1	0

-2-to-1 MUX (module mx2)

1) 입출력

I / O	변수명
Input	d0
	d1
	s
output	y

2) 진리표

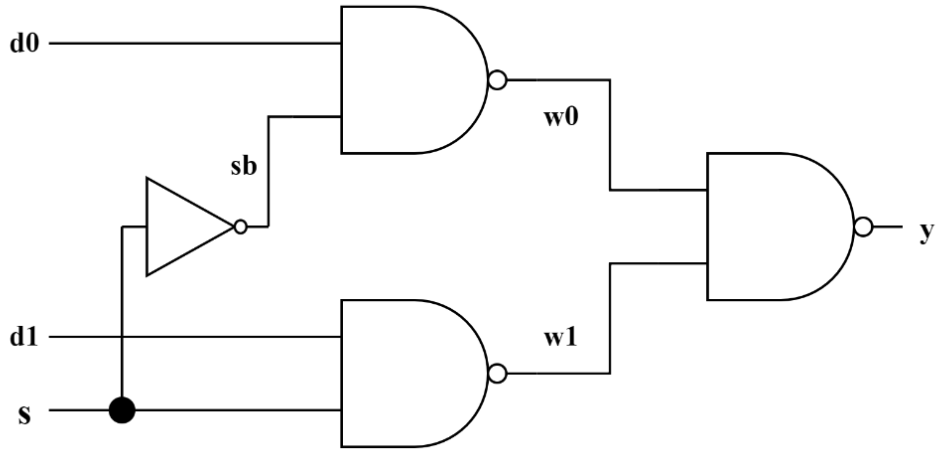
d0	d1	s	y
0	0	0	0
0	1		0
1	0		1
1	1		1
0	0	1	0
0	1		1
1	0		0
1	1		1

3) k-map

s \ d0d1	00	01	11	10
0	0	0	1	1
1	0	1	1	0

➔ 논리식: $y = (\bar{S} \cdot d_0) + (S \cdot d_1)$

4) 회로도



input값인 d0, d1을 s에 연결된 inverter에 따라 두가지 경우로 나누어서 y값으로 출력된다. s가 0일 땐 d0값이, s가 1일땐 d1값이 나오도록 구성되었다. 또한 이들을 이어주는 wire는 각각 sb, w0, s1으로 하여 코드를 짜도록 한다.

4. 설계 검증 및 실험 결과

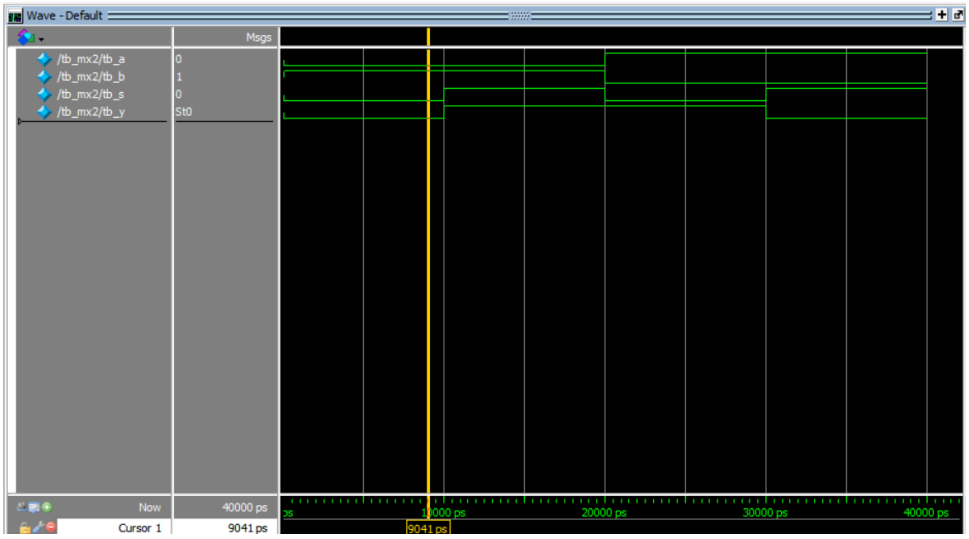
A. 시뮬레이션 결과

-testbench 변수 변화 부분

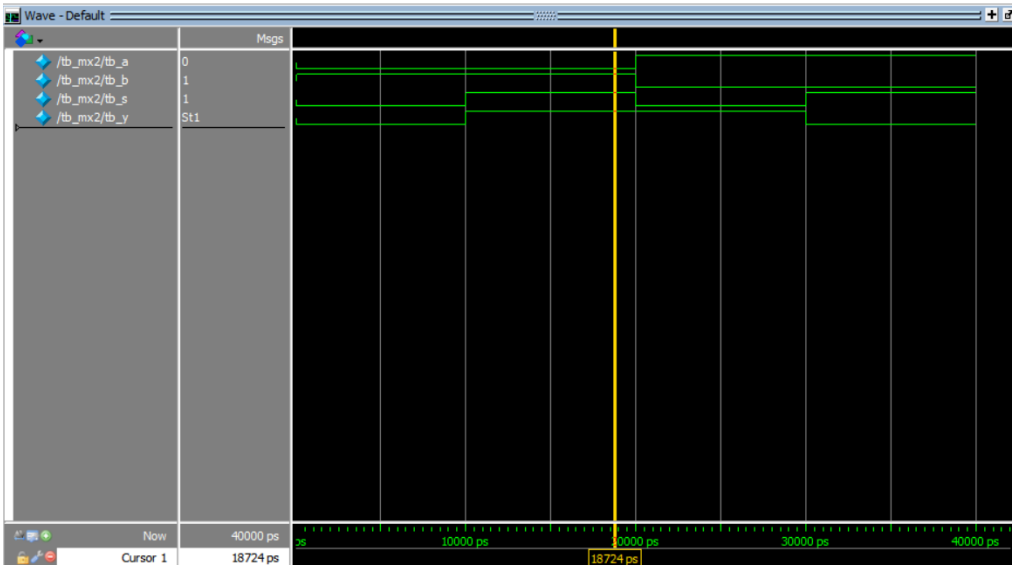
```
        tb_a = 0; tb_b=1; tb_s=0;
#10;    tb_a = 0; tb_b=1; tb_s=1;
#10;    tb_a = 1; tb_b=0; tb_s=0;
#10;    tb_a = 1; tb_b=0; tb_s=1;
#10;
```

-waveform 결과

Input값인 tb_a, tb_b에 먼저 각각 0과 1을 저장한 상태에서 tb_s값이 바뀜에 따라 output인 tb_y의 값이 예상 결과 값과 똑같이 변하는 걸 볼 수 있다. 10ns마다 값이 바뀔 수 있도록 testbench를 구성하여 처음 10ns까지는 tb_s가 0이라 tb_y도 첫번째 input값인 tb_a를 따르고, 10ns이후 20ns전까지 tb_s가 1로 되어서 tb_y도 두번째 값인 1로 바뀌게 된다.



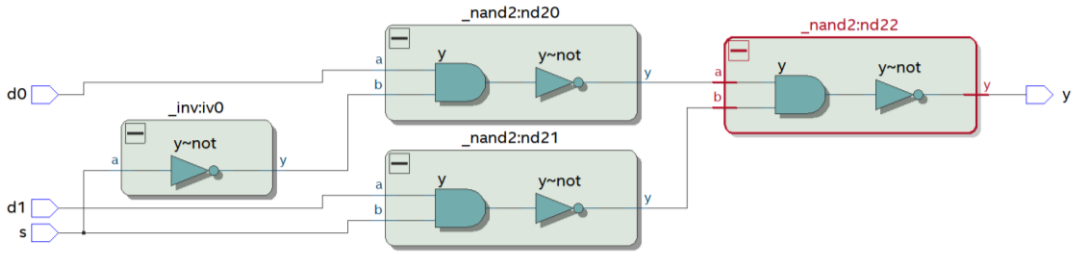
그다음 20ns부터는 tb_a와 tb_b의 값을 서로 바꾸어서 다시 확인한다. input 값 2개가 모두 동일한 경우(tb_a=0, tb_b=0 등)는 결과가 동일하므로 제외시켰다.



B. 합성(synthesis) 결과

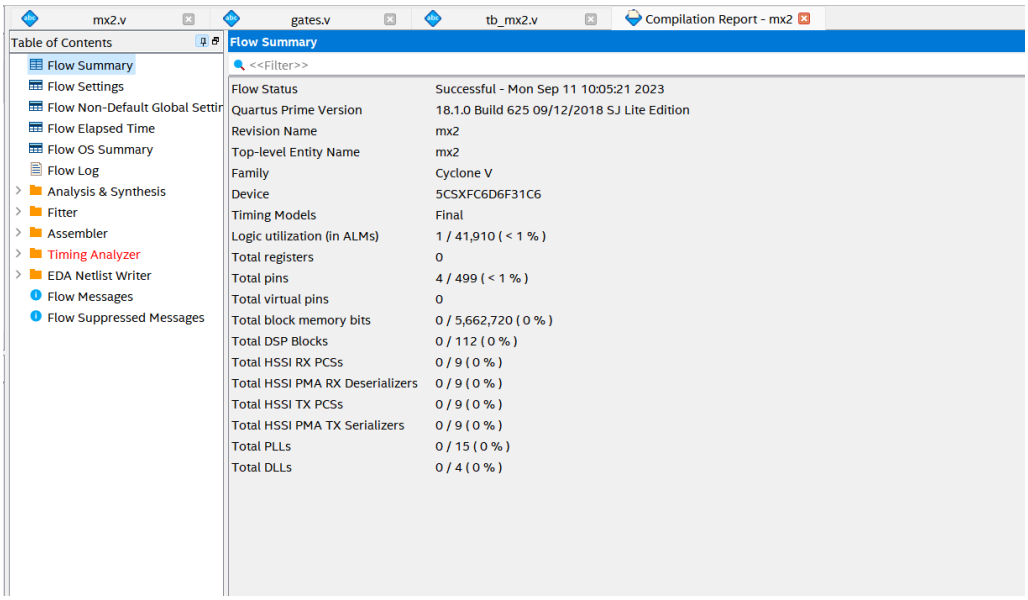
-RTL map viewer

각각의 module로 NAND 게이트와 Inverter 게이트를 구성한 뒤 하나의 MUX module로 합친 결과이다. input값인 d0, d1, s와 wire와 함께 연결되어 마지막 output으로 y값이 나오게 된다.



-Flow summary

mx2를 포함한 파일들을 컴파일한 결과 정상적으로 끝내진 내용을 볼 수 있다.



5. 고찰 및 결론

A. 고찰

프로그램을 돌리면서 문법 오류들이 여러 번 났었는데, 대부분 ';'을 잘못 붙이거나 안 붙여서 생긴 오류들이었다. end나 endmodule에는 안 붙이고, module이름과 테스트벤치때 시간단위에는 붙이는 등으로 맞는 문법으로 고쳐서 해결할 수 있었다. 또한 waveform을 볼 때 테스트벤치에서 쓴 것들이 전부 안 나오고 10ns까지만 나오는 시행착오가 있었는데, 이는 코드 내에서 바꿨던 걸 제대로 저장한 후 다시 돌려서 해결할 수 있었다.

B. 결론

module들을 다른 파일에서 각각 만들고 하나의 mux module로 합치는 방법을 알게되었다. Submodule의 이름과 새로 지은 이름을 쓰고 괄호 안에 '.'을 통해 변수들을 지정하여 연결해주거나, input, output 순서를 고려하여 그대로 쓰는 등의 방식을 더 공부할 수 있었다.

그리고 합치는 쪽인 mux는 top module의 이름과 똑같은 이름으로 module을 선언해줬지만, 다른 sub module들은 top module과 이름이 달라도 상관없다는 것 또한 알게되었다.

6. 참고문헌

David Money Harris and Sarah L. Harris / Digital Design and Computer Architecture / Elsevier / 2007