디지털 시스템 설계 (CSED273) Lab 1. 논리회로 기초 보고서

2023.03.18.

20220923 이윤혁

## 1. 개요

Lab1_1에서는 Xilinx Vivado를 이용해 Verilog의 문법을 익히고, Testbench를 통한 시뮬레이션방법을 익힌다. Lab1_2에서는 Verilog를 이용해 Function set이 Functionally complete set임을 회로를 구성해 증명한다.

## 2. 이론적 배경

### 2.1. Positive Logic / Negative Logic

논리식은 참/거짓 2가지의 값을 가질 수 있다. 이를 전자회로로 구현하기 위해 전자회로의 전압으로 HIGH(5V), LOW(0V)를 사용한다. Positive Logic / Negative Logic은 참/거짓을 HIGH/LOW와 매칭하는 방법이다. Positive Logic은 HIGH를 참과 매칭하고, LOW는 거짓과 매칭한다. Negative Logic은 HIGH를 거짓과 매칭하고, LOW는 참과 매칭한다.

### 2.2. HDL (Hardware description Language)

HDL(Hardware description Language)은 논리회로를 가상공간에 구현하는 프로그래밍 언어이다. 이 과목에서는 Verilog를 사용하여 논리회로를 구현한다.

### 2.3. Functionally complete set

어떤 논리 함수들을 element로 가지는 집합이 모든 불 대수식을 표현할 수 있을 때, 그 집합은 Functionally complete하다. 어떤 집합이 Functionally complete하려면, 그 집합은 element로 가진 함수들을 이용해 AND, OR, NOT 연산을 구현할 수 있어야 한다.

## 3. 실험 준비

Vivado 사용법과 Verilog의 문법을 숙지하고, Testbench의 사용법을 숙지한다.

Lab 1에서는 AND, OR, NOT, NAND, NOR의 연산을 사용한다. 이 연산들의 Truth table은 아래 표와 같다.

| A | B | A AND B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| A | B | A OR B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| A | NOT A |
|---|---|
| 0 | 1 |
| 1 | 0 |

| A | B | A NAND B |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

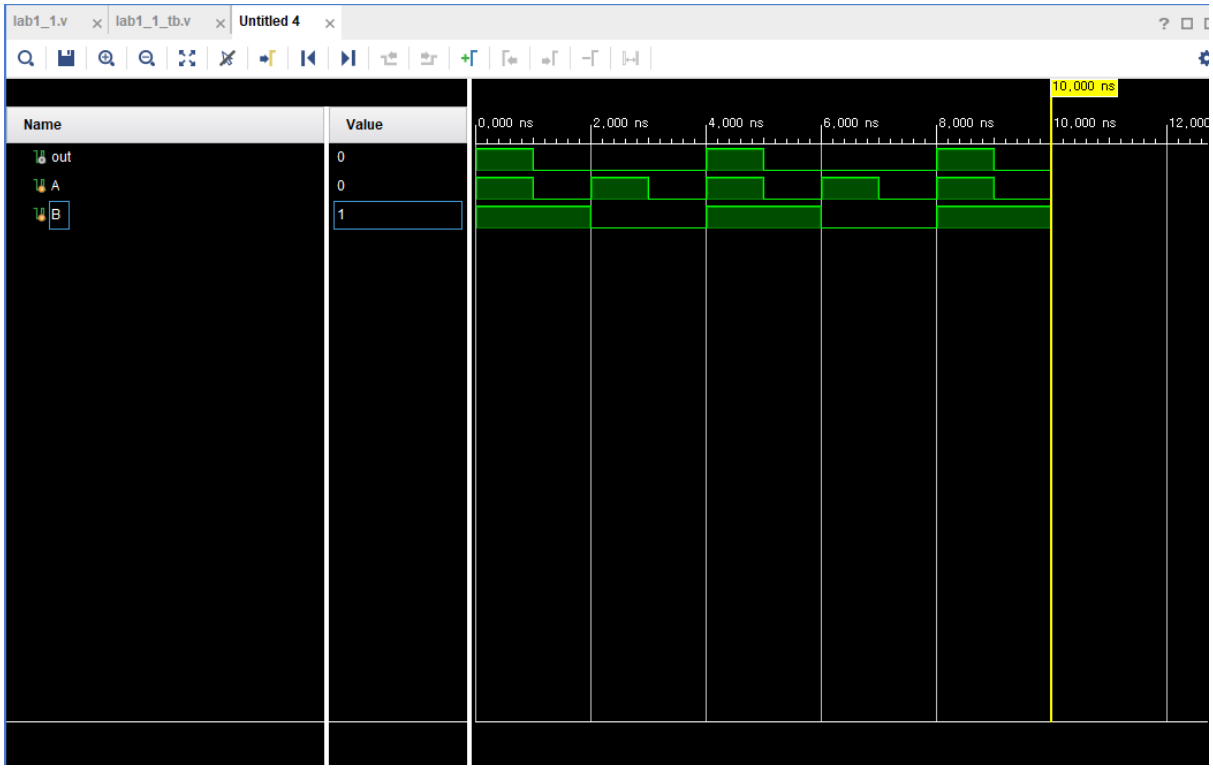
| A | B | A NOR B |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

## 4. 실험 결과

### 4.1. Lab1_1

And 게이트를 Verilog로 구현하여, input의 A를 1ns에 한 번씩, input의 B를 2ns에 한 번씩 반전시키었다. 시뮬레이션 파형은 [그림 1]과 같다. A and B의 결과인 out은 A와 B가 모두 HIGH일 때 HIGH, 그렇지 않을 때 LOW가 나와야 한다. 시뮬레이션으로 파형을 도출한 결과 and 연산이 잘 진행되었다.

**[그림 1] Lab1_1의 시뮬레이션 파형**




4.2. Lab1_2

주어진 function set으로 {AND, OR, NOT}을 구현해 Functionally complete set임을 보였다.

[그림 2], [그림 3], [그림 4], [그림 5]은 각각 {OR, NOT}, {AND, NOT}, {NAND}, {NOR}을 function set으로 {AND, OR, NOT}을 구현한 회로도이다. RTL Analyzer Schematic의 기능을 사용했다.

**[그림 2] Lab1_2.i, {OR, NOT}**

OR과 NOT이 이미 set에 있기 때문에 AND만 구현하면 된다. AND는 [그림 2]와 같이 구현할 수 있으며, Truth table을 이용해 계산해보면 AND와 똑같이 기능한다.

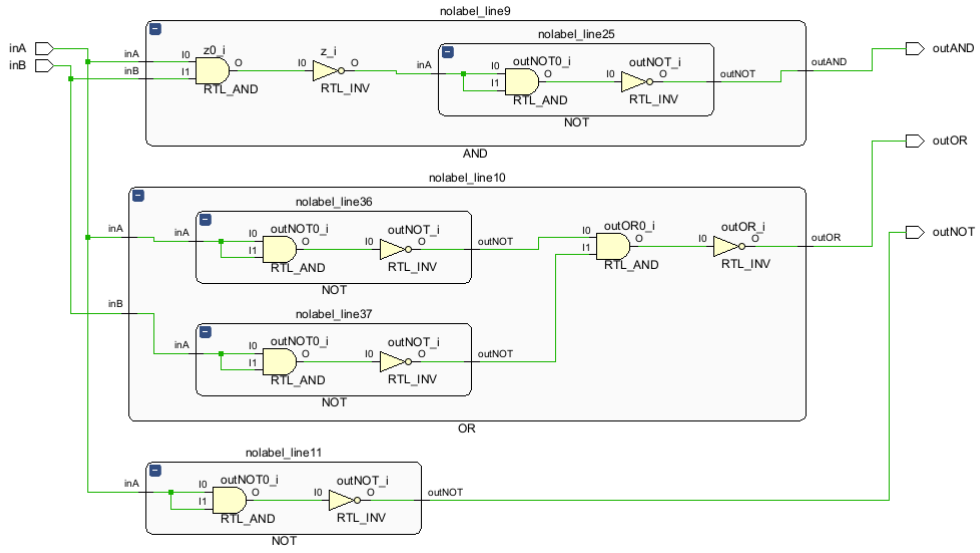
| A | B | NOT A | NOT B | (NOT A) OR (NOT B) | NOT ((NOT A) OR (NOT B)) | A And B |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |

**[그림 3] Lab1_2.ii, {AND, NOT}**


OR과 NOT이 이미 set에 있기 때문에 AND만 구현하면 된다. AND는 [그림 3]와 같이 구현할 수 있으며, Truth table을 이용해 계산해보면 AND와 똑같이 기능한다.

| A | B | NOT A | NOT B | (NOT A) AND (NOT B) | NOT ((NOT A) AND (NOT B)) | A OR B |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |

**[그림 4] Lab1_2.iii, {NAND}**



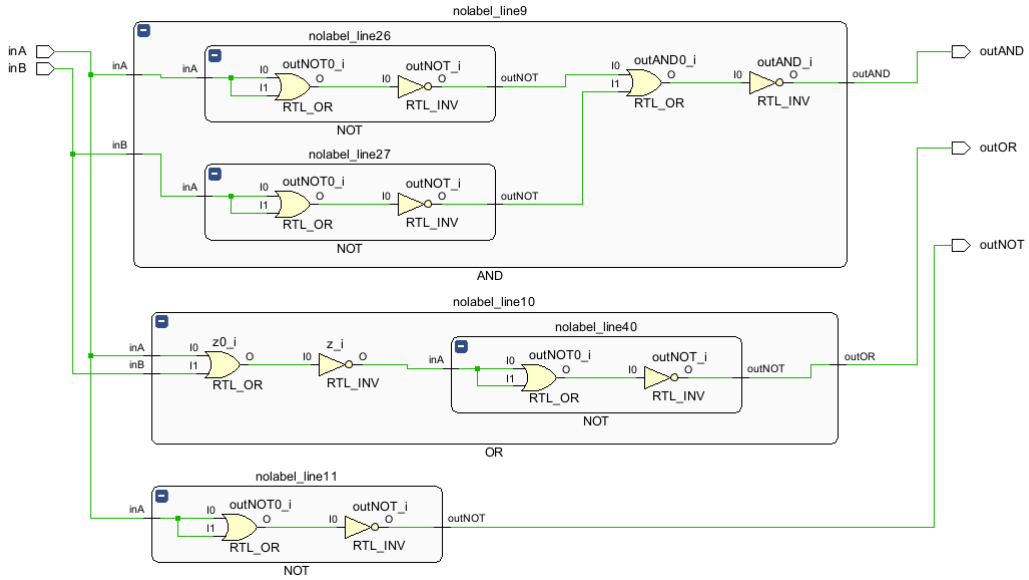

아래 Truth Table로 위 회로도에 구현한 AND, OR, NOT의 회로가 옳음을 증명했다.

| A | A NAND A | NOT A |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 0 |

| A | B | A NAND A | B NAND B | (A NAND A) NAND (B NAND B) | A OR B |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |

| A | B | A NAND B | (A NAND B) NAND (A NAND B) | A AND B |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |

**[그림 5] Lab1_2.iiii, {NOR}**




아래 표로 위 회로도에 구현한 AND, OR, NOT의 회로가 옳음을 Truth Table을 통해 증명했다.

| A | A NOR A | NOT A |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 0 |

| A | B | A NOR A | B NOR B | (A NOR A) NOR (B NOR B) | A AND B |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |

| A | B | A NOR B | (A NOR B) NOR (A NOR B) | A OR B |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |

## 5. 논의

Lab1_1을 해결하며 Verilog로 gate 연산을 하는 방법과 testbench로 input과 output의 simulation을 할 수 있었다. Input의 파형과 연산이 output의 파형과 같은 것을 관찰할 수 있었다.

Lab1_2를 해결하며 주어진 function set로 functionally complete set을 만들 수 있었다. Truth table과 논리 연산을 이용하여 간단해진 AND, OR, NOT 연산을 구현할 수 있었다.

변수를 선언 또는 초기화하고 사용하는 다른 언어들과 다르게, Verilog는 Wire를 선언 또는 초기화 하지 않고 사용하여 실습할 때 어색했으나 실습이 끝난 후 익숙해졌다.