디지털 시스템 설계
Lab 5 – ALU와 JK 플립플롭

2024.05.03.

개요

- 컴퓨터의 기초가 되는 산술 논리 장치(Arithmetic Logic Unit; ALU)와 정보를 저장할 수 있는 JK 플립플롭(Flip-flop)을 구현한다

- 자세한 학습 목표는 다음과 같다.
  - ALU의 산술 및 논리 장치 구현
  - SR 래치와 negative reset master-slave JK 플립플롭 구현
  - 테스트 벤치를 이용한 회로 검증

이론적 배경 – ALU

- ALU는 입력에 대해 여러 산술(Arithmetic) 및 논리(Logic) 연산을 수행한다.

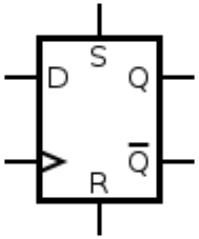
  - 산술 장치
    - 산술 장치는 덧셈, 뺄셈, 곱셈, 나눗셈 등과 같은 수학적 연산을 수행한다.
    - 현대 CPU에는 실수 연산, 삼각 함수 등을 내장하고 있다.

  - 논리 장치
    - 논리 장치는 AND, OR, NOT 등의 Bitwise 논리 연산 등을 처리한다.
    - 복잡한 장치는 Bit shift, bit masking 등등을 지원한다.

이론적 배경 – 비동기 / 동기 회로

- 비동기 회로
  - 클럭을 따르지 않는 순차 회로와 모든 조합 회로가 속한다.

- 동기 회로
  - 클럭을 따르는 순차 회로가 속한다. 신호가 동기화되어 다른 회로와 같은 순간에 맞춰 작동하기 위함이다.

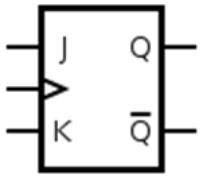


보통 왼쪽 아래와 같이 삼각형으로 표현된 곳이 클럭 입력이다.
출처: https://fr.m.wikipedia.org/wiki/Fichier:D-Type_Flip-flop.svg

이론적 배경 – JK 래치 / JK 플립플롭

- JK 래치는 SR 래치에 추가적인 회로를 더해 S와 R이 동시에 1인 상황에서도 정상적으로 작동하도록 수정한 것이다.
  - JK 래치에서 J와 K가 동시에 1일 경우 현재 상태에 상관없이 값을 반전시킨다.

- 래치가 입력이 바뀔 때 출력도 바로 바뀌는 비동기 회로라면 플립플롭은 입력이 바뀌더라도 출력이 클럭에 맞추어 반영되는 동기 회로이다.
  - JK 플립플롭은 클럭 신호를 추가로 받아 이에 맞추어 작동한다.



JK 플립플롭의 심볼.
출처: https://en.m.wikipedia.org/wiki/File:JK_Flip-flop_%28Simple%29_Symbol.svg

이론적 배경 – Master-slave JK 플립플롭

- Master-slave JK 플립플롭은 SR 래치 두 개를 연결하여 만든 플립플롭이다.
  - 클럭이 1인 동안 Master 래치를 활성화해 입력을 임시로 저장한 뒤 클럭이 0이 되는 순간 Slave 래치로 전달한다.

- 그러나 Master 래치가 활성화 되어있는 동안 글리치로 잠깐 출력 값이 변경되면 다음 클럭이 0이 되는 순간에 Slave 래치로 전파되는 문제가 있다.
  - 이는 클럭이 1인 동안 계속 입력을 받기 때문에 생기는 문제로 클럭이 0에서 1로, 혹은 1에서 0으로 바뀌는 순간에만 입력을 받는 Edge-trigger 회로를 사용하여 해결할 수 있다.

실험에 앞서서...

- 실험 준비

1) 자료에 주어진 표의 값에 따라 ALU를 산술 장치와 논리 장치 두 부분으로 나누어 단순화하고 회로도를 그린다.

2) 두 모듈을 2:1 MUX로 묶어 ALU의 회로도를 그린다.

3) SR 래치의 회로도를 그린다.

4) SR 래치를 사용해 Negative reset Master-slave JK 플립플롭의 회로도를 그린다.

5) SR 래치와 비교해 Master-slave JK 플립플롭이 해결 가능한 글리치와 해결할 수 없는 글리치를 예상하고 분석한다.

* 보고서 필수 내용

- 실험 준비 과정과 회로도

실험에 앞서서...

- 공통 유의사항
  - `/* Add your code here */`부분만 변경하여 구현하기
  - Gate-Level Modeling으로 구현하기

    assign keyword와 bitwise operator 사용 가능

    &&, ||, ! 는 논리 연산자로 비트 연산자가 아님에 유의

    +, -, * 등 산술 연산자 사용시 0점 처리
  - 모든 입출력은 Little-endian 형식으로 표현한다.

실험에 앞서서...

- 모든 입출력은 Little-endian 형식으로 표현한다.

```
wire [7:0] C; //Little endian [7] [6] [5] [4] [3] [2] [1] [0]
wire [0:7] D; //Big endian    [0] [1] [2] [3] [4] [5] [6] [7]
```

```
module lab4_2(
    input [4:0] in_a,
    input [4:0] in_b,
    input in_c,
    output [4:0] out_s,
    output out_c
    );
```

출처: https://electronics.stackexchange.com/questions/267510/confusion-over-wire-array-notation-in-verilog

실험

- lab5_1 – ALU

  1. 산술 장치와 논리 장치 모듈을 구현하고 이를 사용해 ALU를 완성한다.
  2. Schematic 기능으로 회로를 확인한다.

- lab5_1_tb – ALU 테스트 벤치

  1. 테스트 벤치를 완성하고 시뮬레이션을 실행해 정상 작동을 확인한다.

실험

- lab5_2 – Master-slave JK 플립플롭

1. SR 래치 모듈을 구현하고 이를 사용해 Negative reset Master-slave JK 플립플롭을 완성한다.
2. Schematic 기능으로 회로를 확인한다.

- lab5_2_tb – Master-slave JK 플립플롭 테스트 벤치

1. 테스트 벤치를 완성하고 시뮬레이션을 실행해 주어진 조건에서 정상 작동을 확인한다.
2. 테스트 벤치를 수정해 SR 래치 대비 해결된 글리치와 해결되지 못한 글리치를 모두 보인다.

실험

- 보고서 필수 내용
  - Schematic 기능으로 생성한 회로도 캡처
  - 시뮬레이션 파형 캡처

제출 방법

- (코드) {학번}_lab5.zip으로 다음 파일을 압축해 PLMS로 제출하기
  - lab5_1.v
  - lab5_1_tb.v
  - lab5_2.v
  - lab5_2_tb.v
  - lab5_report.pdf