EE312.<Introduction to Computer Architecture> Lab1 Report

20170616 정희진
20140727 이동현

1.Introduction

ALU(arithmatic logic unit)는 컴퓨터 cpu에서 숫자의 논리연산을 담당하는 부분이다. lab1은 베릴로그를 이용해 ALU를 구현해보며 베릴로그의 기본 문법을 익혀보는 것을 목적으로 한다. 우리가 구현해야 하는 ALU는 2개의 16비트 2진수의 논리연산을 할 수 있는 ALU로, 2개의 16비트 이진수 A,B 그리고 operation의 종류를 결정해주는 4비트 이진수 OP 이렇게 총 3개의 input을 받는다.

구현해야 하는 operation은 덧셈, 뺄셈, or, xor, shift등을 포함한 16개인데 덧셈과 뺄셈에서 overflow를 검출하기 위하여 Cout을 register에 저장해주었으며, Cout이 1인경우 overflow가 있는 것으로 보았다. C는 A와 B의 연산 결과이고 이 역시 register에 저장해주었다. 따라서 output은 C, Cout 두가지가 된다.

2.Design

ALU를 구현하기 위해 하나의 모듈만을 이용하였으므로 figure는 따로 그리지 않았다. always @(*)와 case를 이용하여 OP값에 따라 16가지 케이스를 분류해주었으며 3항 조건 연산자 ? : 를 이용하여 overflow를 검출해주었다.

3.Implementation

arithmatic right shift의 경우 signed bit 값이 필요하므로 올바르게 해주기 위하여 input과 output에 해당하는 wire에 signed를 추가해주었다.
덧셈에서 overflow가 나오는 case는
i)양수와 양수를 더했는데 음수가 나오는 경우
ii)음수와 음수를 더했는데 양수가 나오는 경우
이렇게 두가지가 있으며 뺄셈에서 overflow가 나오는 case는
i)양수에서 음수를 뺐는데 음수가 나오는 경우 tight
ii)음수에서 양수를 뺐는데 양수가 나오는 경우
이렇게 두가지가 있다. 이런 경우에는 Cout값을 1로 만들어줘야 하므로 A(15), B(15), Cadd wire, Csub wire를 참조하여 Cout 값을 1로 만들어주었다.

4.Evaluation

모든 testcase를 통과하였으며 속도도 빨랐다.

5.Discussion

덧셈과 뺄셈의 경우 충분한 testcase가 존재하지 않아 overflow 부분에서 오타를 냈음에도 fail이 뜨지 않았습니다. 코드를 읽어보던 도중 오타를 찾아 수정하긴 했지만 테스트케이스에서 발견됐으면 더 좋았을 것 같습니다.

6.Conclusion

베릴로그의 기본 문법과 modelsim을 이용해 컴파일하고 시뮬레이션 하는 방법을 다시 remind할 수 있었습니다.