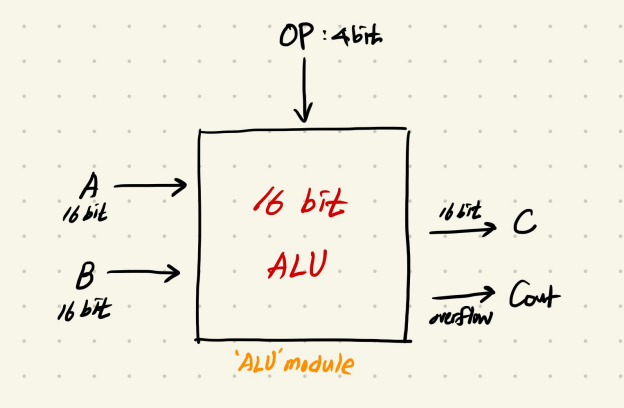
EE312 Computer Architecture

LAB1 Report

20160542 이현지, 20160707 형준하

1. Introduction
    - 이번 LAB에서는 signed 16 bit signed binary input 2개 A and B를 넣어 16 bit signed binary output C and overflow를 detect하는 Cout을 배출하는 ALU를 만들어보는 목표하에 간단한 Verilog 문법과 ModelSim 사용법을 익히는 시간을 가져보았다.
    - ALU라는 module 안에 case문을 통해 어떤 operation을 진행할지 결정하는 4 bit OP 값에 따라 해당 operation을 진행할 수 있도록 작성하였다. Add and Sub의 경우는 overflow 여부를 확인하는 Cout도 따로 관리하였고, 나머지의 경우는 0으로 입력하였다. 16 bit reg K and 1 bit Kout을 따로 선언하고 이를 always 외부에서 C and Cout에 assign하고 operation 결과 값이 always 내부에서 K and Kout에 저장되는 방식으로 코드를 작성하였다. 대체적인 경우, 해당 operation에 대한 Verilog syntax가 존재하여 이를 이용해 코드를 간단하게 만들고자 했다.

2. Design
    - 이번에는 간단한 ALU를 만드는 과정이였기에 하나의 모듈, ALU 안에서 모든 과정이 다 진행되었다. ALU module을 도식화 하면 아래와 같이 나온다.

- 

[Figure of implementation]

3. Implementation
    - case문을 통해 OP값에 따라 총 16개의 operation 중에 무엇이 진행될지 정해주었고, 16 bit signed input A and B가 해당 operation을 진행한 output을 C and Cout으로 내보내었다. C는 16 bit signed output이고 Cout의 경우는 add and sub operation 진행할 때 overflow 여부가 일어나면 1, 아니면 0이 나오는 값이다. 나머지 operation의 경우는 0으로 나오게 하였다.
    - overflow가 진행되는 경우를 각각 확인해본 결과, Add의 경우는 A, B의 MSB 값은 같고, 더한 결과 값인 K의 MSB는 다른 경우 그리고 Sub의 경우는 B와 K의 MSB 값이 같고 A의 MSB 값이 다른 경우에 해당하였다. 이에 알맞게 if문으로 이 경우들에 Kout = 1으로 값을 선언해주었고, 나머지의 경우 Kout의 default 값인 0으로 나오도록 하였다.
    - always 내에 wire assignment가 불가하여 register K and Kout을 선언하고 여기에 operation 결과값을 저장하고 각 register을 C and Cout wire에 연결시켜서 그 값이 나가는 방식으로 진행하였다.
    - 대체적인 operation의 경우는 그에 해당하는 Verilog syntax가 이미 존재하여 이를 사용하여 코드를 간소화하였다.

4. Evaluation
    - 50개 중 50개 다 pass 했습니다.

- 이번 과제의 경우 대다수의 operation이 이미 verilog에 있는 syntax를 그냥 가져다가 쓰는 경우가 많아서 그런 부분에 대해서는 크게 evaluate 방법을 고민하지 않았습니다.
- Cout의 값이 바뀔 수도 있는 Add and Sub operation의 경우에 대해서는 여러가지 경우들의 값이 맞는지 확인하면서 evaluation을 진행했습니다.

5. Discussion
    - 이번 랩에서 Verilog를 처음 써보았기에 언어 자체의 문법이나 나오는 에러들이 낯설어서 많이 헤맸습니다. 또한 Cout을 처음에 Carry Out 값으로 이해하여 나오는 fail 경우들을 고치는데 어려움을 겪었습니다.

6. Conclusion
    - 이번 Lab1을 통해 ModelSim와 Verilog을 처음 사용해보면서 주로 사용했던 high-level language와의 차이점을 비교하는 시간을 가졌고, 여러가지 operation들을 다시 익히는 시간을 가졌다. 또한, Verilog의 여러가지 특성들을 이번 랩에서 에러들을 고치면서 찾아볼 수 있었다.