Verilog를 이용한 mini_cpu 설계

과 목 디지털 논리회로

전기공학부
20192066 한지윤 (팀장)
20192043 전민태 (팀원)
교류학과
20246121 한주헌 (팀원)

목 차

1. 프로젝트 목표

프로젝트 목표

1. HDL(Hardware Description Language) 언어에 대한 이해를 높인다.
2. CPU의 동작 원리를 학습한다.
3. Verilog를 사용하여 간단한 ALU 동작을 구현하는 mini CPU를 설계한다.
4. Place & Route 과정을 통해 결과를 분석한다.

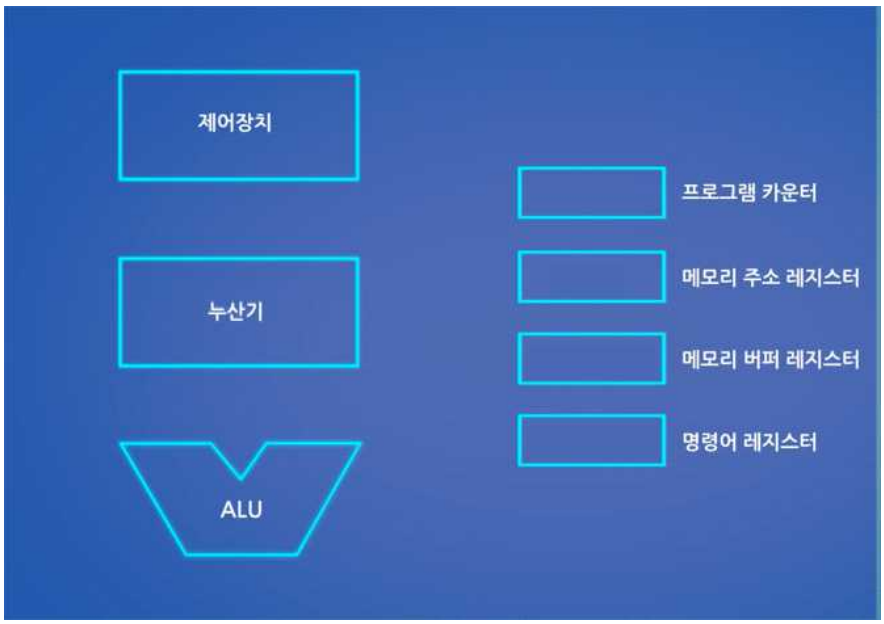
2. 프로젝트 구성

CPU (Central Processing Unit)

가. CPU 구성

Mini_CPU는 다음과 같은 구성 요소들로 이루어져 있다.

- Program Counter (PC): 다음에 실행할 명령어의 주소를 저장한다.
- Memory Address Register (MAR): PC에서 주소를 받아 해당 주소의 데이터를 가져온다.
- Memory Buffer Register (MBR): MAR에서 가져온 데이터와 명령어를 일시 저장한다.
- Instruction Register (IR): MBR에 저장된 데이터 중 명령어를 저장한다.
- Arithmetic Register (AR): 연산에 사용될 데이터를 저장한다.
- 제어장치: 명령어를 해독하고 실행한다.
- ALU (Arithmetic Logic Unit): 산술 및 논리연산을 수행한다.
- Memory : 데이터 및 명령어를 저장한다.



<그림 1 : CPU 구조>

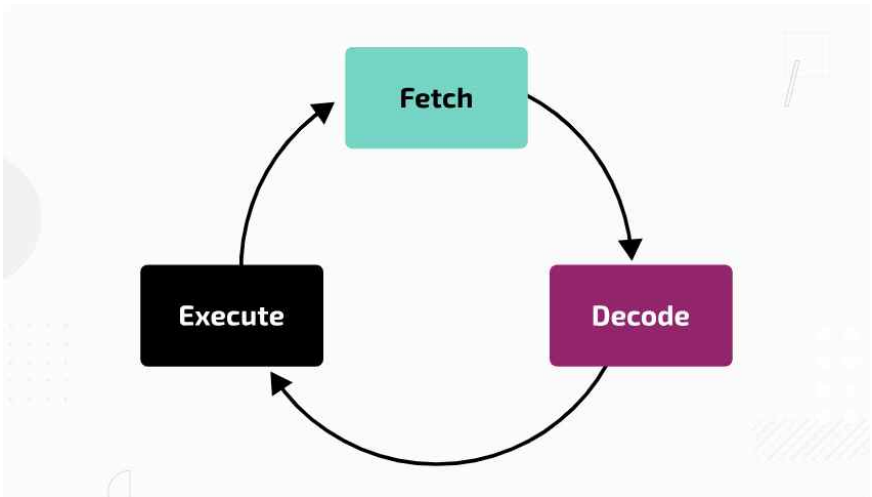
나. CPU 동작

CPU는 크게 Fetch(인출) - Decode(해석) - Execute(실행) 세 단계로 나타낼 수 있다.

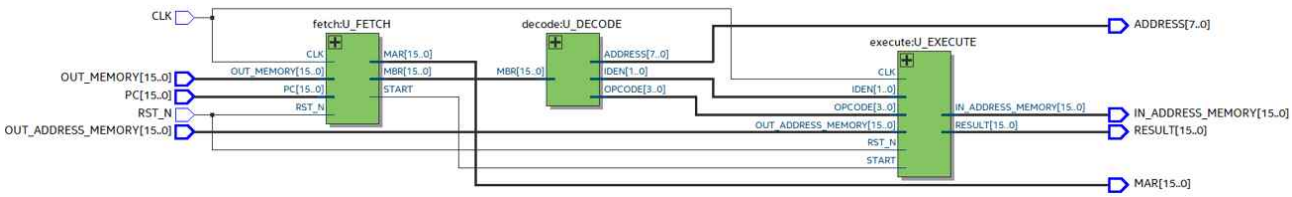
- Fetch : 실행할 명령어를 가져오는 단계
MAR에서 PC에서 주소를 넘겨받아 데이터 및 명령을 받아오고 MBR로 저장하는 과정이다.

- Decode : 명령어를 해석하는 단계
제어장치가 저장된 명령어를 해석하고, 해당 명령어를 실행하는 데 필요한 시스템 자원에 지시를 보내는 과정이다.

- Execute : 실행단계 (ALU 실행)
해석된 명령어를 실제로 실행하는 단계. 이 단계에서 ALU와 다른 레지스터들이 동원되어 명령어에 따른 연산을 수행한다.



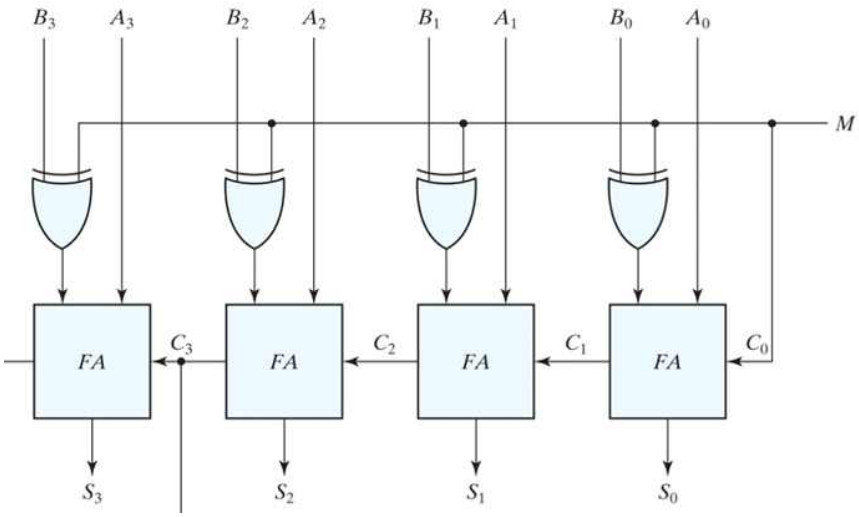
<그림 2 : CPU 동작 과정>



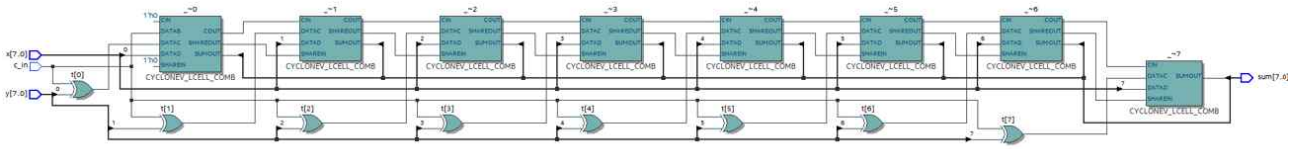
ALU (Arithmetic Logic Unit)

가. Adder/Subtractor

- Full_Adder를 기반으로 CRA(Carry Ripple Adder)방식으로 설계
- M(c_in) = 0 : B와 xor연산을 하면 B가 그대로 나옴 --> ADD
- M(c_in) = 1 : B와 xor연산을 하여 B가 toggle(1의 보수) + 1(2의 보수)로 계산 -> SUB
- 8bit 연산을 하기 위해 최종 캐리는 무시하는 방향으로 진행



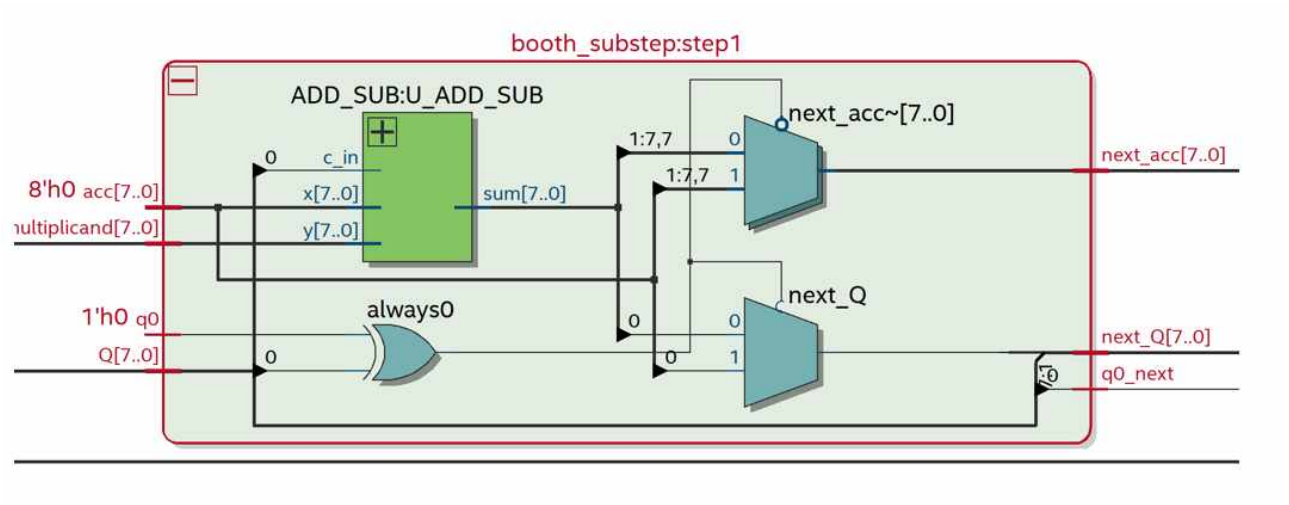
<그림 4 : ADD/SUB 구조>



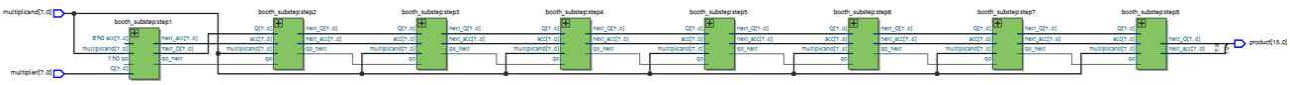
<그림 5 : 8bit ADD/SUB RTL>

나. Multiplier

- Booth 알고리즘을 이용하여 곱셈기 설계
- Booth 알고리즘 : 곱하는 수의 1이 연속으로 나오면 덧셈 연산을 그만큼 진행하여 연산 속도가 느려진다. 이를 방지하기 위해서 Booth 알고리즘을 사용하였다.
- 8bit * 8bit 결과로 16bit결과가 나온다.



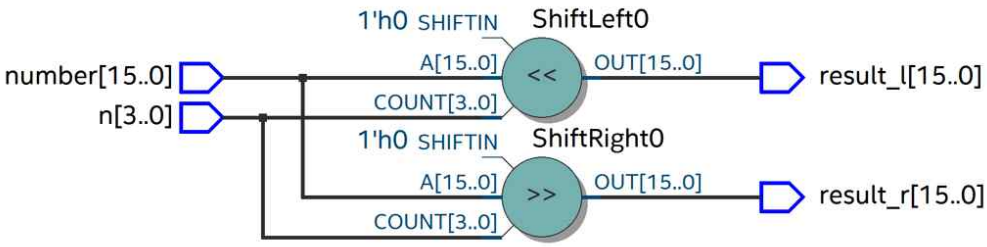
<그림 6 : 8 * 1 Booth Multiplier>



<그림 7 : 8 * 8 Booth Multiplier>

다. Barrel Shifter

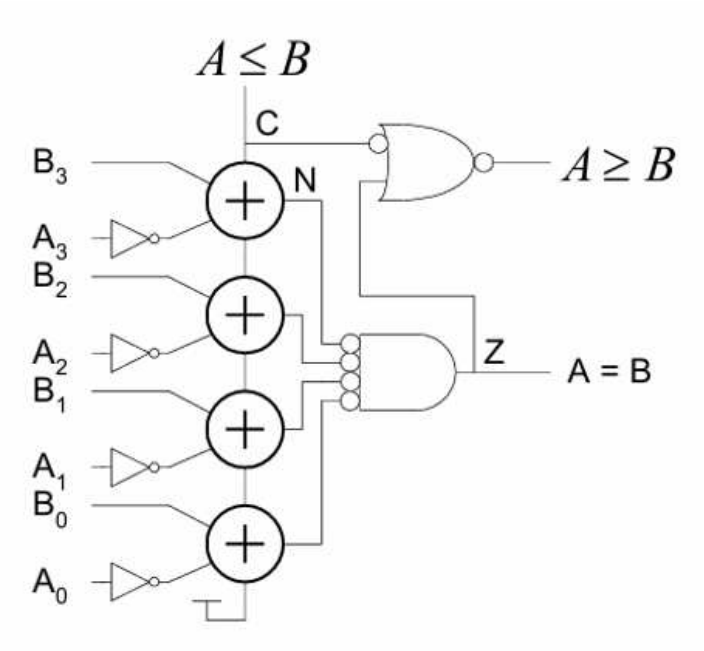
- Shifter와의 차이점은 N개의 비트를 한 번에 시프트를 시킬 수 있다는 점이다.
- Logical shift로 설계 (shift이후 0으로 채움, 시프트 1번 당 *2와 같음)



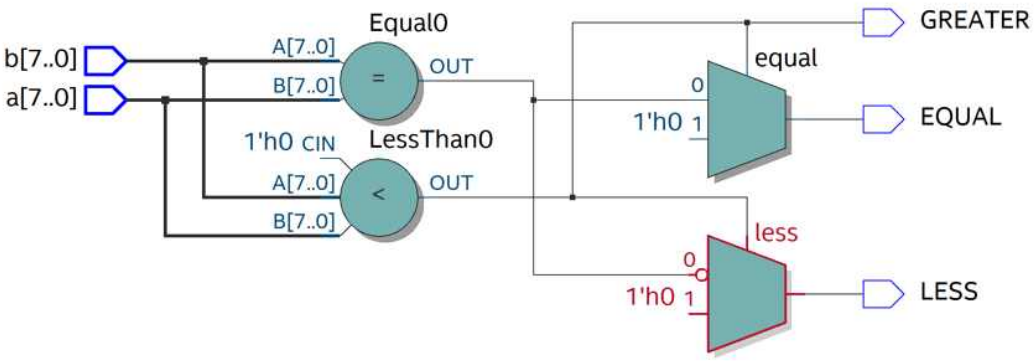
<그림 8 : Barrel Shifter RTL>

라. Comparator

- 최상위 비트를 빼서 크고 작음을 비교
- B-A를 했을 때 Carry가 발생하면 B가 큼
- 최상위 비트가 같으면 다음 상위 비트를 계산하는 방식을 반복



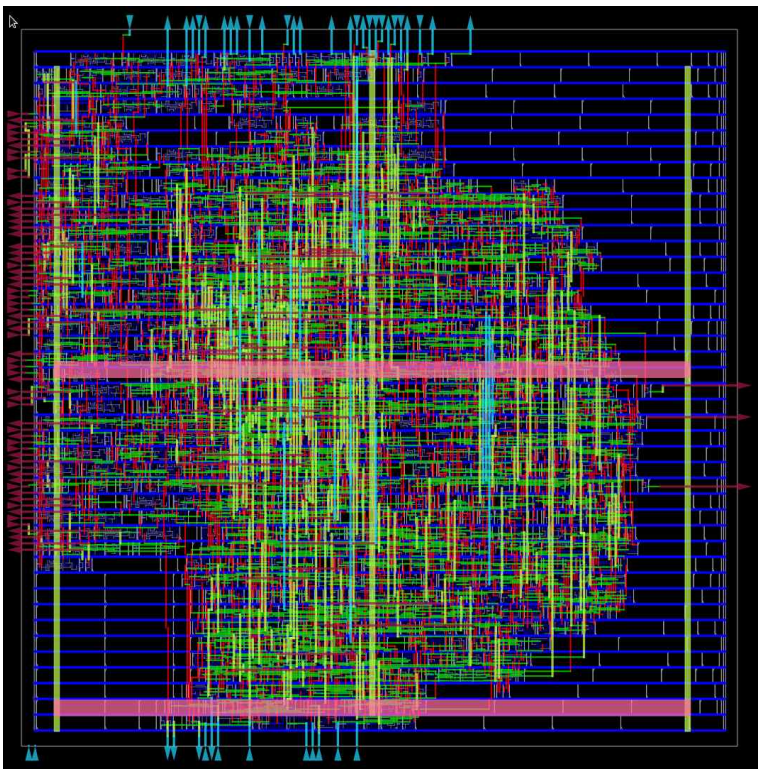
<그림 9 : Comparator>



<그림 10 : Comparator RTL>

Place & Route

- OpenRoad : VLSI 설계의 검증 및 테스트를 위한 도구
- 회로의 내부 칩 구성, 크기, 전력 소모량 및 Worst Slack을 확인하기 위해 사용
- Nangate45 (Nangate사의 45nm 공정)에서의 결과를 분석
- Place & Route 결과는 세세하게 확인하려 했지만, 한 모듈의 소자들이 떨어져 있어 모듈의 확인이 어려움
- Area = 2250u^2, 61% Util
- Totla_Power = 1.72e-2W
- WNS = 0 --> time period가 넉넉하지만 줄여도 0으로 나와 멈췄다.



<그림 11 : Place & Route 결과>

```
================================================================================
finish report_power
--------------------------------------------------------------------------------
Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power (Watts)
----------------------------------------------------------------
Sequential             3.09e-04   2.65e-04   7.43e-06   5.82e-04   3.4%
Combinational          8.69e-03   7.87e-03   4.32e-05   1.66e-02  96.6%
Clock                  0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
Macro                  0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
Pad                    0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
----------------------------------------------------------------
Total                  9.00e-03   8.13e-03   5.06e-05   1.72e-02 100.0%
                          52.4%      47.3%       0.3%

================================================================
finish report_design_area
----------------------------------------------------------------
Design area 2250 u^2 61% utilization.
[WARNING GUI-0076] QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
Elapsed time: 0:03.37[h:]min:sec. CPU time: user 1.98 sys 0.33 (68%). Peak memory: 195156KB.
```

```
================================================================================
finish report_tns
--------------------------------------------------------------------------------
tns 0.00

================================================================================
finish report_wns
--------------------------------------------------------------------------------
wns 0.00

================================================================================
finish report_worst_slack
--------------------------------------------------------------------------------
worst slack INF
```

<그림 12 : Report & Log 파일>

3. 프로젝트 수행 과정

가. CPU의 동작 과정 이해

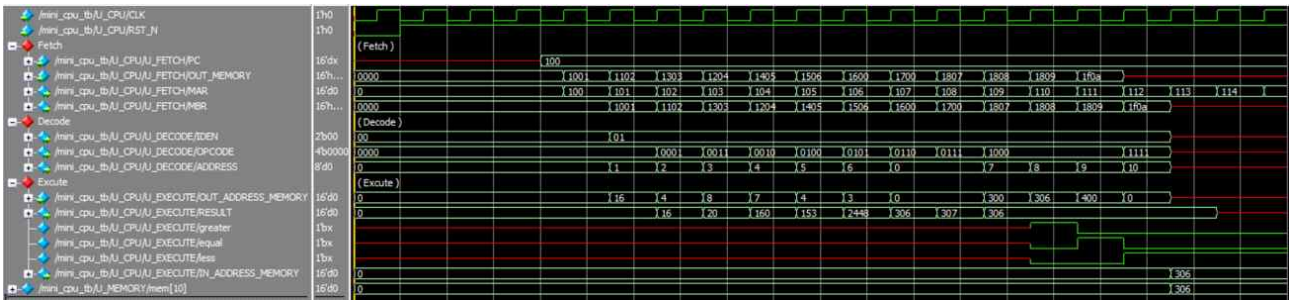
CPU의 인출-해석-실행 과정을 이해하고, Verilog를 이용해 간단한 CPU를 설계해보았다.

- 16bit의 명령어(0000_OPCODE_regA_regB)가 들어오면 OPCODE를 해석하여 regA, regB에 들어있는 값을 실행하는 방식으로 설계하며 CPU의 동작 원리를 이해했다.
- 과제를 수행하며 CPU의 동작 과정을 이해하는데 도움이 되었다.

나. 세 단계로 나누어 CPU 설계

인출, 해석, 실행과정에 해당하는 Fetch, Decode, Execute 모듈을 각각 설계하고 하나의 CPU 모듈로 인스턴스화 시켜 CPU를 설계했다.

- 메모리 모듈에서 16bit 데이터(명령/데이터 구별_OPCODE_주소/데이터)를 불러와 CPU의 동작 원리에 따라 직관적으로 실행되는 CPU를 설계하고 동작을 확인했다.



<그림 13 : 전체 시뮬레이션 결과>

```
// instruction
mem[100] = 16'b0001_0000_0000_0001; // ar = 16
mem[101] = 16'b0001_0001_0000_0010; // ar = 16 + 4 = 20
mem[102] = 16'b0001_0011_0000_0011; // ar = 20 * 8 = 160
mem[103] = 16'b0001_0010_0000_0100; // ar = 160 - 7 = 153
mem[104] = 16'b0001_0100_0000_0101; // ar = 153 << 4 = 153 * 16 = 2448
mem[105] = 16'b0001_0101_0000_0110; // ar = 153 >> 3 = 153 / 8 = 306
mem[106] = 16'b0001_0110_0000_0000; // ar = 306 + 1 = 307
mem[107] = 16'b0001_0111_0000_0000; // ar = 307 - 1 = 306
mem[108] = 16'b0001_1000_0000_0111; // great = 1
mem[109] = 16'b0001_1000_0000_1000; // equal = 1
mem[110] = 16'b0001_1000_0000_1001; // less = 1
mem[111] = 16'b0001_1111_0000_1010; // mem[10] = 306
```

<그림 14 : Memory 및 시뮬레이션 과정>

다. ALU 구성 요소 추가

- 처음에는 ADD, SUB, Shift만 넣어서 작동을 확인하였다.
- 추가로 필요한 곱셈기, 비교가 등을 추가하여 ALU 설계를 마쳤다.

라. OpenRoad를 이용한 Place & Route

- 설계한 CPU의 내부 구조, 크기, 전력 소모량 및 Worst Slack을 확인하기 위해 P&R을 실행하였고, GUI, Report파일과 Log파일을 통해 확인하였다.

4. 수행 과정 및 역할 분담

가. 역할 분담

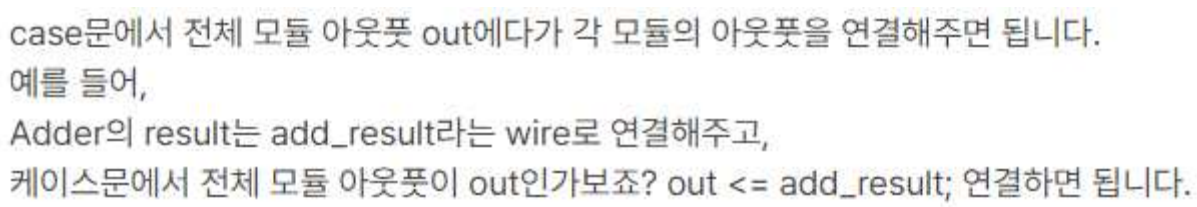
- 한지윤 : 전반적인 CPU 설계 및 마무리, P & R 결과 분석.
- 전민태 : 자료 조사 및 ALU 기초 설계, 테스트 벤치 실행, P & R 결과 분석.
- 한주헌 : 자료 조사 및 보고서 초안 작성.

나. 오류 해결

1) 코드 오류 해결

대부분 오류는 코드를 작성하는 데서 발생했다. 예를 들면 오타, 문법 오류, 인스턴스 불가 등으로 간단하게 에러 메시지를 확인하며 해결할 수 있었다.

가장 중요했던 오류는 인스턴스화 시켜서 출력을 내보낼 때 출력이 나오지 않는 현상이 발생하였는데, 이 문제는 질문을 통해 출력 wire가 하나로 인식되어 에러가 뜬 것을 확인할 수 있었고, 출력을 여러 개로 나누어 하나를 선택하여 출력하는 방식으로 수정했다.



<그림 15 : 질문에 대한 답변>

5. 참고문헌

bRd 3D. "CPU는 어떻게 작동할까?" YouTube, 2021, https://www.youtube.com/watch?v=Fg00LN30Ezg&t=777s
https://github.com/The-OpenROAD-Project