10주차 결과보고서

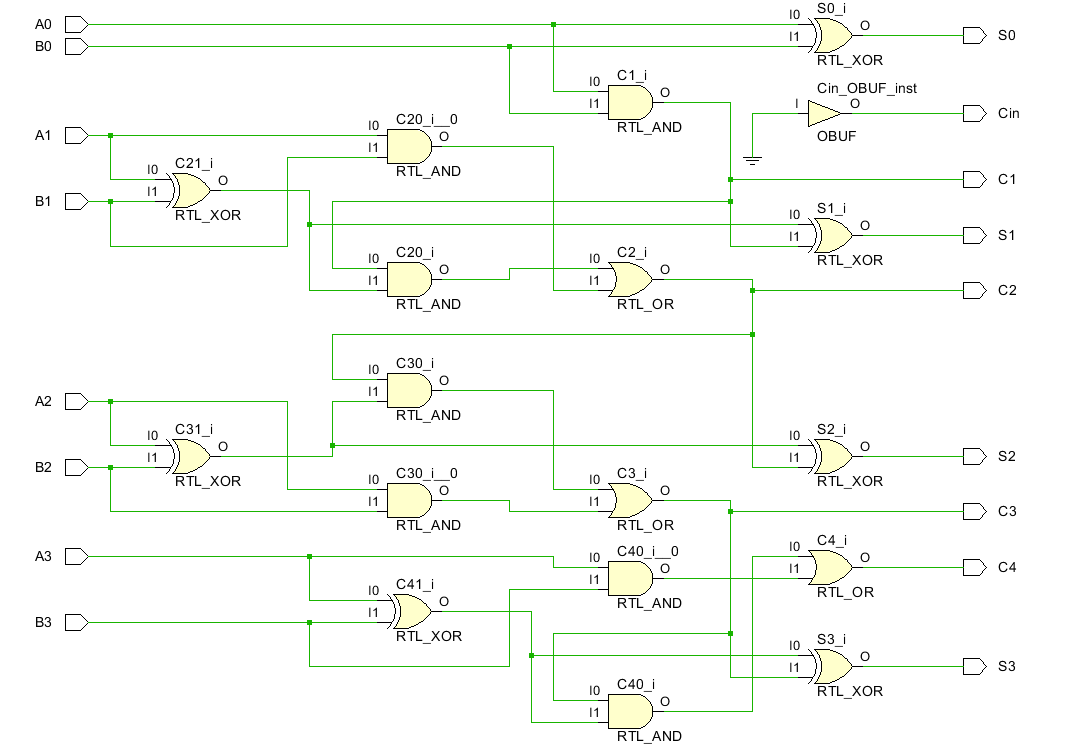
전공 : 국제한국학과 학년 : 4학년 학번 : 20181202 이름 : 김수미

**1. 4bit Binary Parallel Adder 의 결과 및 Simulation 과정에 대해서 설명하시오. (verilog source, 출력 예시, 과정 상세히 적을것)**

1) Source Code

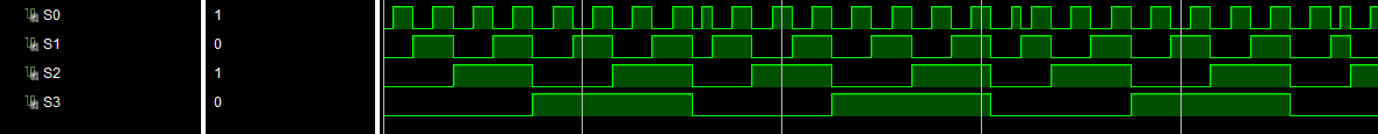
|  |  |
| --- | --- |
| Design Source | Test Bench |
| 텍스트, 영수증이(가) 표시된 사진  자동 생성된 설명 | 텍스트, 영수증이(가) 표시된 사진  자동 생성된 설명 |

2) Schematic



3) Simulation

텍스트, 녹색이(가) 표시된 사진

자동 생성된 설명  


4) 결과 및 과정

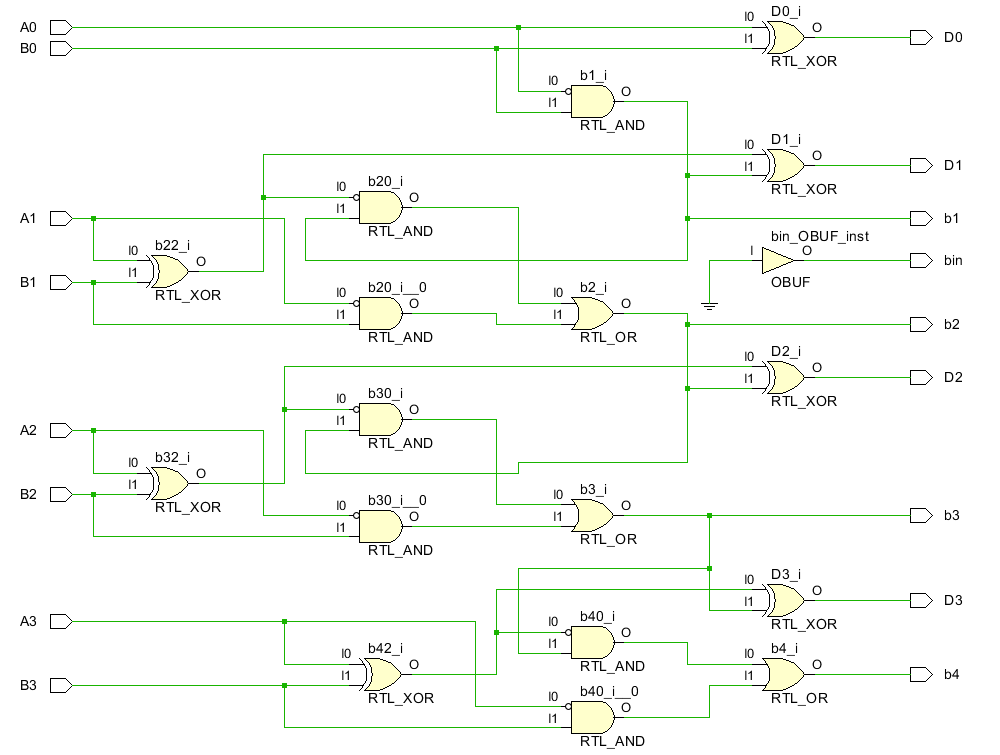
4 bit binary parallel adder는 1bit full adder 4개를 parallel하게 연결한 것이다. 즉 각 bit 자릿수 마다 adder 연산을 수행하며, 발생하는 sum 값은 결과로, carry out 값은 다음 자릿수 adder의 carry in 으로 전달해주면 된다. Verilog 코드는 이와 같은 방식으로 작성해 주었고, FPGA 동작과 simulation 결과를 통해 잘 작동하는 것을 확인했다.

**2. 4bit Binary Parallel Subtractor 의 결과 및 Simulation 과정에 대해서 설명하시오. (verilog source, 출력 예시, 과정 상세히 적을것)**

1) Source Code

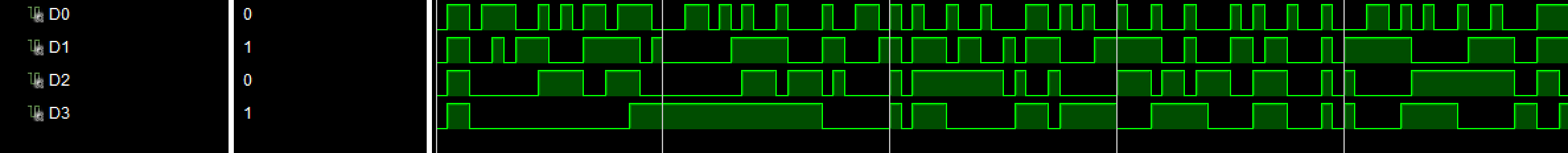
|  |  |
| --- | --- |
| Design Source | Test Bench |
| 테이블이(가) 표시된 사진  자동 생성된 설명 | 텍스트, 영수증이(가) 표시된 사진  자동 생성된 설명 |

2) Schematic



3) Simulation

텍스트이(가) 표시된 사진

자동 생성된 설명  


4) 결과 및 과정

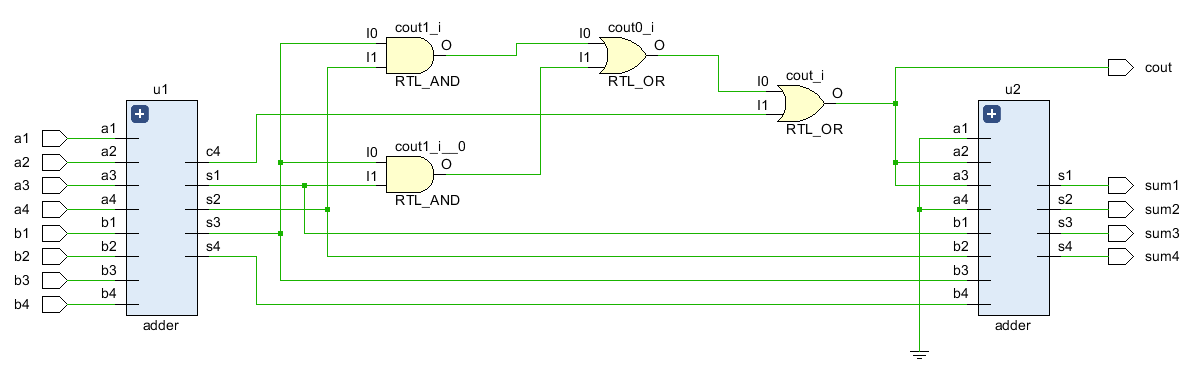
4 bit binary parallel subtractor는 1bit full subtractor 4개를 parallel하게 연결한 것이다. 즉 각 bit 자릿수 마다 subtractor 연산을 수행하며, 발생하는 difference 값은 결과로, borrow out 값은 다음 자릿수 subtractor의 borrow in 으로 전달해주면 된다. Verilog 코드는 이와 같은 방식으로 작성해 주었고, FPGA 동작과 simulation 결과를 통해 잘 작동하는 것을 확인했다.

**3. BCD Adder 의 결과 및 Simulation 과정에 대해서 설명하시오. (verilog source, 출력 예시, 과정 상세히 적을것)**

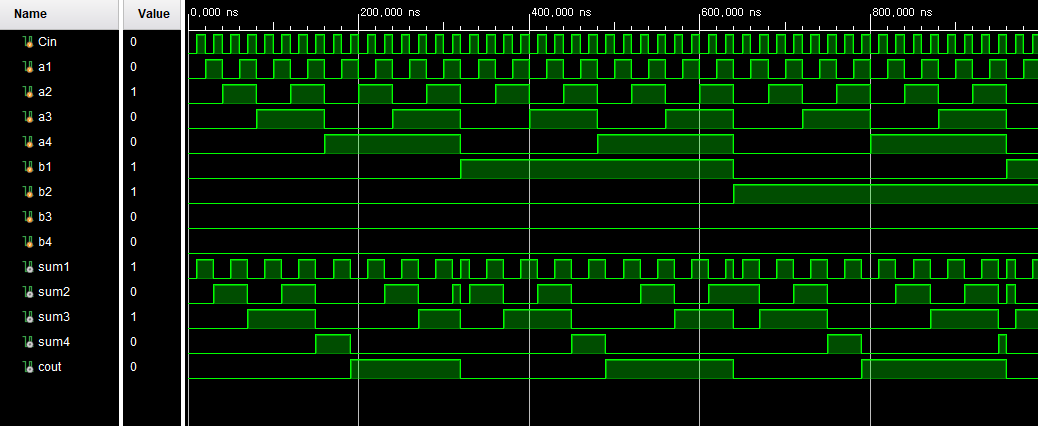
1) Source Code

|  |  |
| --- | --- |
| Design Source | Test Bench |
|  | 텍스트, 영수증이(가) 표시된 사진  자동 생성된 설명 |

2) Schematic



3) Simulation



4) 결과 및 과정

실습 시간에 구현한 BCD Adder는 4bit adder의 연산 결과를 BCD 표현 방식으로 변환해 주는 장치이다. 연산 결과 값이 9보다 큰 수인 경우 6(0110)을 더해 주기만 하면 된다. 이를 구현하기 위해 먼저 4bit adder 연산을 수행하는 회로 하나와, 연산 결과가 9 이상의 수인 경우 0110을 첫번째 연산 결과에 더해주는 회로(보상회로), 즉 총 두개의 adder회로를 이용하여 장치를 구성해 주었다.

**4. 결과 검토 및 논의 사항**

처음 4bit의 add 연산 결과가 9보다 큰 수인지를 check 해주는 bit를 연산하는 과정에서 변수를 한 자리씩 밀려서 연결해주는 바람에 아주 오랜 시간을 허비했다. 계속 코드와 회로도 그림을 보면서 디버깅했지만 찾을 수 없어서 결국 하나씩 값을 대입해 계산하며 디버깅을 함으로써 오류를 찾을 수 있었는데, c언어에서 연산 결과 값을 print 해가면서 디버깅을 하는 것 처럼, 귀찮더라도 연산 결과 값을 simulation에서 확인하면서 디버깅 하는 것의 중요성을 느낄 수 있었다.

**5. 추가 이론 조사 및 작성**

1) Verilog의 Design Source에서 wire변수를 활용하는 방법

Verilog 언어로 코드를 작성할 때, input 또는 output으로 지정하지 않고 변수를 사용할 수 있는 방법이 있다. 바로 wire type으로 변수를 선언하는 것이다. 굳이 입력 또는 출력으로 사용자가 입력해주거나, 출력 값을 확인할 필요가 없는 값들을 이런 wire 변수로 선언해줄 수 있다. 예를 들어 BCD adder에서 처음 adder의 연산 결과는 굳이 값을 출력해줄 필요가 없다. 우리가 원하는 값은 처음 adder의 연산 결과에 보상 회로 연산을 적용한 결과값이기 때문이다. 따라서 처음 adder의 연산 결과 값처럼 굳이 변수를 simulation 등에서 출력해볼 필요가 없는 경우, 이는 output 변수로 선언할 필요 없이 wire 변수로 선언해주면 된다.

|  |
| --- |
| Design Source에서 wire 변수의 사용 예시 |
|  |

2) Verilog의 Design Source에서 module을 활용하는 방법

마치 c언어에서 함수를 사용하는 것처럼, 이미 선언해 놓은 module을 가져다가 사용할 수 있는 방법이 있다. 먼저 아래의 사용 예시를 살펴보자.

|  |  |
| --- | --- |
| Design Source에서 module 호출 사용의 예시 | |
|  |  |

하나의 파일 안에서 adder module과 BCD module 두개를 선언했다. 그 다음 BCD module 안에서 adder u1(~) 식으로 adder module을 호출해서 사용한 것을 확인할 수 있다. 이와 같은 방식으로 이미 선언해 놓은 module을 가져다가 사용하면 같은 코드를 여러 번 쓸 필요 없이 깔끔하게 코드를 작성할 수 있다.