13주차 결과보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20212022 이름: 이예준

**1.**

스크린샷, 텍스트, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*4-bit Shift Resister Simulation 결과

위의 사진은 Shift Resister를 Simulation한 결과이다. 이번에 구현한 Shift Resister는 마지막

Flipflop에 데이터를 입력 받아 Clock Signal에 맞춰Resister내에 있는 각각의 Flipflop이

자신이 저장하고 있는 데이터를 자신 앞에 있는 Flipflop으로 Shift하는 기능을 가지고 있다.

엄밀히 말하면, 앞에 있는 Flipflop으로 출력을 내보내고, 뒤에 있는 Flipflop의 데이터를

받아 저장하는 것이다.

clk는 Clock Signal이며, clk가 1이 될 때마다 Flipflop이 그 Signal을 받아 Shift Resister의

기능을 하게 된다. D는 데이터를 뜻하며, D가 1이 되면 Shift Resister의 첫번째 Flipflop의

입력을 들어간다. rst는 Reset기능을 하며, Active Low로 작동하기 때문에 rst가 0일 때

Reset기능이 활성화되어 Resister내에 있는 모든 데이터를 초기화하며, rst가 1일 때는

Reset기능이 비활성화된다. 위의 Simulation 결과를 보게 되면 clk이 1이 될 때마다 Shift

Resister의 데이터가 한칸씩 옆으로 Shift하는 것을 볼 수 있고, D가 1일 때는 데이터가

Output[3] 위치에 추가되는 것을 볼 수 있다. 마지막으로 rst가 0일 때 초기화가 정상적으로

되는 것을 볼 수 있다. 아래 사진은 Shift Resister의 Verilog 코드와 Simulation 코드이다.

텍스트, 전자제품, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*4-bit Shift Resister Verilog 코드

이 코드를 통해 위에서 봤던 Simulation 결과를 얻을 수 있었다. 이 코드는

4-bit Resister이기 때문에 output Out은 [3:0] 배열로 나타냈으며, , clk, D 그리고 rst는

input으로 선언해주었다. posedge를 이용해 clk가 1이 될 때마다 begin~end 문을

수행하도록 했다. 만약 rst가 0이면, Out은 0으로 초기화하며, 그 외에는 Out의 데이터를

오른쪽으로 Shift한다. 따라서 데이터 D가 왼쪽에서 오른쪽으로 Shift되기 위해 Out[3]에

입력된 D가 추가된다. simulation 코드에서는 clk를 10클럭마다, D는 50클럭마다

반전시켜주었고, rst는 400클럭마다 반전시켜주어 rst가 0일 때의 Out결과와 rst가

1일 때의 D가 입력될 때마다 달라지는 Out결과를 볼 수 있게 해주었다. 예를 들어

첫번째 Clock Signal에 rst가 1이고 D가 1이라면, Out은 10002이 되고,

두번째 Clock Signal에 rst가 1이고 D가 0이라면, Out은 01002이 되고,

세번째 Clock Signal에 rst가 1이고 D가 1이라면, Out은 10102이 되고,

네번째 Clock Signal에 rst가 0이고 D가 1이라면, Out은 00002이 된다.

**2.**

스크린샷, 디스플레이, 텍스트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*4-bit Ring Counter Simulation 결과

위의 사진은 Ring Counter를 Simulation한 결과이다. 이번에 구현한 Ring Counter는 처음에

첫번째 Flipflop의 초기값을 1을 저장해두고 Clock Signal에 맞춰 Counter내에 있는 각각의

Flipflop이 자신이 저장하고 있는 데이터를 자신 앞에 있는 Flipflop으로 Shift하는 기능을

가지고 있다. Shift Resister와의 차이점은 마지막 Flipflop에 저장되어 있는 데이터를 그냥

버리는 것이 아니라 첫번째 Flipflop의 입력으로 출력을 내보내서 마지막 Flipflop의 데이터를 다시 첫번째 Flipflop으로 돌려보낸다는 점이다. 데이터가 계속 순회하기 때문에

Ring이라는 이름이 붙었다.

clk는 Clock Signal이며, clk가 1이 될 때마다 Flipflop이 그 Signal을 받아 Ring Counter의

기능을 하게 된다. rst는 Reset기능을 하며, Active Low로 작동하기 때문에 rst가 0일 때

Reset기능이 활성화되어 Counter내에 있는 모든 데이터를 초기화하며, rst가 1일 때는

Reset기능이 비활성화된다. 위의 Simulation 결과를 보게 되면 clk이 1이 될 때마다

Ring Counter의 데이터가 한칸씩 옆으로 Shift하는 것을 볼 수 있고, 데이터가 배열 끝까지

가면 다시 배열 시작지점으로 돌아오는 것을 확인할 수 있다. 마지막으로 rst가 0일 때

초기화가 정상적으로 되는 것을 볼 수 있다.

아래 사진은 Ring Counter의 Verilog 코드와 Simulation 코드이다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*4-bit Ring Counter Verilog 코드

이 코드를 통해 위에서 봤던 Simulation 결과를 얻을 수 있었다. 이 코드는

4-bit Ring Counter이기 때문에 output Out은 [3:0] 배열로 나타냈으며, , clk과 rst는

input으로 선언해주었다. posedge를 이용해 clk가 1이 될 때마다 begin~end 문을

수행하도록 했다. 만약 rst가 0이면, Out은 0으로 초기화하며, 만약 Out이 10002이 되면

다시 데이터를 처음 위치로 Shift하기 위해 Out을 00012로 저장한다. 그 외의 경우는

우리가 구현한 Ring Counter에서 초기값으로 Out[0]에 데이터를 저장했기 때문에,

Out의 데이터를 왼쪽으로 Shift한다. simulation 코드에서는 clk를 10클럭마다, D는

50클럭마다 반전시켜주었고, rst는 400클럭마다 반전시켜주어 rst가 0일 때의 Out결과와

rst가 1일 때의 D가 입력될 때마다 달라지는 Out결과를 볼 수 있게 해주었다.

예를 들어 rst가 1이면 Clock Signal에 맞춰서

00012 → 00102 → 01002 → 10002 → 00012 → 00102 이런 과정을 거치며,

rst가 0이 되면, Out은 00012이 된다(1을 남겨두지 않고 0으로 만들면 데이터를

추가시키는 방법을 구현하지 않았기 때문에 Ring Counter의 기능을 하지 못한다).

**3.**

스크린샷, 텍스트, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*4-bit Up/Down Counter Simulation 결과

위의 사진은 Up/Down Counter를 Simulation한 결과이다. 이번에 구현한 Up/Down Counter는 기존에 더하는 기능을 하는 Up Counter와 빼는 기능을 하는 Down Counter의

두 기능을 합친 Counter로, D값이 0이면 Up Counting을 하고, D값이 1이면

Down Counting을 하는 Counter이다. 또한 현재 상태가 Up이냐 Down이냐에 따라

7-segment Display을 이용해 표시해준다.

clk는 Clock Signal이며, clk가 1이 될 때마다 Flipflop이 그 Signal을 받아 Ring Counter의

기능을 하게 된다. rst는 Reset기능을 하며, Active Low로 작동하기 때문에 rst가 0일 때

Reset기능이 활성화되어 Counter내에 있는 모든 데이터를 초기화하며, rst가 1일 때는

Reset기능이 비활성화된다. 또한 D는 현재 Counter가 Up상태인지 Down상태인지

알려주는 값으로 0이면 Up상태이고, 1이면 Down상태이다. Seg는 7-segment Display를

나타내는 배열로 각 segment 수에 맞게 [6:0]배열로 나타낸다. 위의 Simulation 결과를 보게

되면 clk이 1이 될 때, D가 0이면 Out값이 증가하고, D가 1이면 Out값이 감소하는 것을

볼 수 있고, rst가 0일 때는 초기화가 정상적으로 되는 것을 볼 수 있다.

아래 사진은 Ring Counter의 Verilog 코드와 Simulation 코드이다.

텍스트, 전자제품, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

\*4-bit Up/Down Counter Verilog 코드

이 코드를 통해 위에서 봤던 Simulation 결과를 얻을 수 있었다. 이 코드는 4-bit Up/Down

Counter이기 때문에 output Out은 [3:0] 배열로 나타냈으며 7-segment를 나타내는 Seg는

[6:0] 배열로 나타냈다. D, clk과 rst는 input으로 선언해주었다. posedge를 이용해 clk가

1이 될 때마다 begin~end 문을 수행하도록 했다. 만약 rst가 0이면, Out은 0으로

초기화한다.

도표, 텍스트, 평면도, 스크린샷이(가) 표시된 사진

자동 생성된 설명 , 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명 , 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

\*7-segment Display, alphabet D, U 그림

Seg[0]는 g segment와 연결되어 있고, Seg[1]은 f segment와 연결되어 있다.

따라서 만약 D가 0이면 Seg[0] = 0, Seg[1] = 1이 되고, Out값에 1을 더한다.

만약 D가 1이면 Seg[0] = 1, Seg[1] = 0이 되고, Out값에 1을 뺀다. 또한 Seg 초기값은

‘4b1111000으로 되어 있는데, 현재 코드에서 a segment는 아예 안 쓰기 때문에 Seg[2]는

a segment와 연결되어 있다.

simulation 코드에서는 clk를 10클럭마다, D는 200클럭마다 반전시켜주었고, rst는 400클럭마다 반전시켜주어 rst가 0일 때의 Out결과와 rst가 1일 때의 D가 입력될 때마다 달라지는 Out결과를 볼 수 있게 해주었다. 예를 들면,

Clock Signal에 맞춰서 D가 0이면

Display에는 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명을 나타내고 있으며,

00012 → 00102 → 00112 → 01002이 되며,

Clock Signal에 맞춰서 D가 1이면

Display에는 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명 을 나타내고 있으며,

01002 → 00112 → 00102 → 00012 이 된다.

rst가 0이 되면, Out은 00002이 된다.

**4.**

이번 실험을 통해 Shift Resister와 Ring, Up/Down Counter에 대해 Verliog를 이용한

구현 방법에 대해 알아보고 Simulation결과와 FPGA 보드 구현 결과가 정상적으로 나오는

것을 확인할 수 있었다. Shift Resister는 Clock Signal에 맞춰 Flipflop의 데이터가 정해진

방향으로 이동하는 것을 볼 수 있었고, 사용자가 데이터를 입력하면 다음 State에 데이터가

추가되는 것을 볼 수 있었다. 또한Flipflop의 끝까지 간 데이터는 버려지는 것도 볼 수

있었다. Ring Counter는 Shift Resister와 같이 Clock Signal에 맞춰 Flipflop의 데이터가

정해진 방향으로 이동하는 공통점을 가지고 있었지만, 끝까지 간 데이터를 다시

첫 Flipflop의 입력으로 넣어주어 Loop를 돌게 하는 차이점도 있었다.

Up/Down Counter는 사용자에 따라 데이터를 더하거나 빼는 Mode로 바뀔 수 있었다.

이번 실험의 논의 사항으로 Ring Counter에 대한 보완점을 생각해보았다. 이번 실험에서

구현한 Ring Counter는 단순히 데이터를 순회하는 기능을 가지고 있어 Reset을 해도

데이터를 모두 초기화하는 것이 아니라 첫번째 Flipflop에 데이터를 남겨서 순회를 하게

만들었다. 따라서 Shift Resister와 같이 데이터를 입력하는 기능을 추가하고 Reset을

할 때는 데이터를 모두 초기화하는 기능으로 수정하면 좋을 것이다.

**5.**

Counter를 설계할 때는 몇가지 과정을 거치게 되는데,

첫번째로 할 것은 설계하고자 하는 Counter의 특징과 기능들을 정한다. 두번째로 할 것은

Counter의 State Graph 또는 State Diagram을 그린다.

원이(가) 표시된 사진

자동 생성된 설명

\*State Diagram

State Graph를 가지고 각 입력이 Flipflop의 입력에 따라 어떤 출력을 보이는지 알 수 있는

State Table을 그린 뒤 State Table을 토대로 각 Flipflop의 입력에 대한 카르노 맵을 그린다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

\*State Table

도표, 텍스트, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

\*Karnaugh Map

카르노 맵을 통해서 Characteristic Equation을 구한 뒤 그 식으로 회로도를 그리면 완성된다.

도표, 평면도, 라인, 기술 도면이(가) 표시된 사진

자동 생성된 설명

\*Counter 회로도