컴퓨터 공학 기초 실험2 보고서

실험제목: Latch & flip-flop design with/without reset/set

실험일자: 2021년 10월 11일 (월)

제출일자: 2021년 10월 17일 (일)

학 과: 컴퓨터정보공학부

담당교수: 공진흥 교수님

실습분반: 월요일 0, 1, 2

학 번: 2018202046

성 명: 이준휘

1. 제목 및 목적
   1. 제목

Latch & flip & flop design with/without reset/set

* 1. 목적

해당 수업을 통해 이전 값을 유지하고 있는 저장 소자 역할을 하는 latch와 flip-flop을 설계하고 더불어 reset 과 set 기능을 구현한다. 또한, 구현한 flip-flop을 사용하여 N-bits regiser를 구현하는 방법에 대하여 살펴보도록 한다.

1. 원리(배경지식)
   * 1. D latch

D latch는 CLK이 0일 때에는 기존의 기억하고 있는 값 Q를 저장하고 있고, CLK이 1일 때 들어오는 신호 D의 값을 Q값으로 바꾸는 회로다. D latch는 기본적으로 SR latch를 기반으로 한다. SR Latch에 들어가는 신호 중 S와 R 신호를 CLK과 D, CLK과 D’의 AND 연산 신호로 보내어 SR Latch에서 기존에 불가능했던 신호인 S와 R이 1일 경우를 배제할 수 있도록 하고, CLK을 통해 기존 값을 통제할 수 있도록 하였다.

* + 1. D Flip-Flop

D Flip-Flop은 CLK의 rising edge나 falling edge에서만 D 값으로 출력이 바뀌게 된다. 다른 경우에도 D 값이 바뀌더라도 Q값을 그대로 유지한다.

해당 회로는 D Latch 2개를 통해 구성할 수 있다. 우선 하나의 D latch의 q를 다른 D latch의 D로 연결한다. 그 후 D 값이 들어오는 latch에는 CLK’의 값을, Q를 D값으로 받는 latch에는 CLK 값을 연결해준다.

해당 회로대로 설계할 경우 만약 CLK이 0일 경우에는 앞쪽의 latch는 CLK’이 1로 들어가고 있으므로, Q에 D 값이 바뀌어서 출력되고 있다. 하지만 뒤쪽의 latch에서는 CLK 값이 0임으로 데이터를 받지 못하고 있다. 이 때 CLK이 1로 변하면 뒤쪽의 latch는 CLK이 1로 변하여, 기존에 저장하고 있던 이전 Q의 데이터가 출력될 수 있다. 하지만 앞의 latch에는 CLK’이 0이기에 기존의 Q값을 저장할 뿐 D값은 이에 영향을 미치지 못한다.

위와 같은 방식은 rising edge에 동작하는 Flip-Flop이고, 만약 CLK의 NOT gate를 반대로 연결할 경우 이는 falling edge D Flip-Flop이 된다.

* + 1. enable D Flip-Flop

해당 회로는 D Flip-Flop에서 만약 en의 신호가 1일 경우에만 d를 출력하고 0일 경우에는 기존 값을 저장하는 회로다.

* + 1. Resettable / Settable D Flip-Flop

Resettable / Settable D Flip-Flop은 D flip-flop에 reset과 set 기능이 추가된 D filp-flop이다. reset과 set은 active low, 즉 reset이나 set 값이 0일 때 기능을 수행한다. reset이 0일 경우 Q는 무조건 0을, set이 0일 경우 reset이 0일 경우를 제외하고 1을 출력하도록 한다.

* + 1. N-bits register

해당 회로는 N개의 register를 한 줄로 늘어놓음으로써 N개의 비트를 저장할 수 있는 회로다.

* + 1. Async/Sync Set/Resettable D flip-flop

해당 회로는 Set과 Reset이 Asyncronous 또는 Syncrounous하게 동작하는 회로다. Async의 경우에는 Set 또는 Reset이 동작하면 그 즉시 회로의 output이 CLK에 상관없이 바뀌게 된다. 하지만 Sync의 경우는 하지만 Set 또는 Reset이 동작하면 CLK이 rising edge로 바뀔 때 값이 변화하게 된다. 즉 CLK값이 결과에 영향을 미치게 된다.

1. 설계 세부사항
   * 1. dlatch

해당 파일에는 D Latch를 구현하는 파일이다. 해당 회로를 만들기 위해 Gate를 활용하는 것이 아닌 조건문을 활용하여 제작되었다. 언제나 d나 clk 값이 변할 때 작동할 수 있도록 always문을 걸어주었고, if 문을 통해 만약 clk이 1일 경우만 q에 d 값이 들어갈 수 있도록 만들었다. 마지막으로 q\_bar는 언제나 Q의 invert된 값을 가지도록 만들었다.

* + 1. \_dff

해당 파일에서는 D Flip-Flop을 구현하는 파일이다. 우선 clk의 신호의 invert된 신호를 사용하기 때문에 gates.v의 \_inv 모듈을 사용하여 clk을 여기에 연결한다. rising edge D Flip-Flop을 구현하기 때문에 첫 번째 D latch에는 clk\_bar와 d를 연결해주고, 두 번째 latch에는 clk과 첫 번째에서 나온 w\_q를 연결시켜준다.

* + 1. \_dff\_en

해당 모듈은 enable D flip-flop을 구현하기 위해 제작되었다. 기존에 제작된 mx2를 이용해 q와 d를 입력으로 받고 en에 의해 선택되는 MUX를 연결한다. 그 후 \_dff 모듈을 MUX에서 나온 신호와 연결하여 \_dff\_en 회로를 구현한다.

* + 1. \_dff\_r

해당 모듈은 r(reset\_n)의 값에 의해 신호를 초기화 할 수 있는 Resettable D Flip-Flop이다. 해당 모듈에서는 기존 gates.v에서 만든 \_and2 모듈을 활용하여 d와 reset\_n을 합성한 신호를 만든다. 그 후 \_dff 모듈에 해당 신호를 연결하여 reset\_n이 1일 때만 d의 신호를 받고 이외의 경우에는 0으로 초기화 하는 active low reset을 구현한다.

\_dff\_rs

해당 모듈은 r(reset\_n)과 s(set\_n)의 값에 의해 신호를 초기화 할 수 있는 set/resettable D Flip-Flop이다. 해당 모듈은 d 신호와 ~set\_n 신호를 gates.v 파일에서 만든 OR 연산으로 묶는다. 그 후 해당 신호를 \_dff\_r과 같은 방식으로 연결한다. 해당 방식으로 연결하게 되면 set\_n이 0일 때는 1을, reset\_n이 0일 때는 0을, 둘 다 0일 때는 reset\_n의 우선순위가 높기 때문에 0을 출력하는 회로가 완성된다.

* + 1. \_register8

해당 모듈은 \_dff를 사용하여 8-bit Register를 구현하였다. 해당 회로는 각각의 자리의 d와 q를 clk과 함께 \_dff에 연결하여 만들었다.

* + 1. \_register32

해당 모듈은 \_register8을 사용하여 32-bit Register를 구현하였다. 해당 회로는 각각의 8비트의 자리의 d와 q를 clk과 함께 \_register8에 연결하여 만들었다.

* + 1. \_dff\_rs\_sync

해당 모듈은 Syncronous D Flip-Flop을 제작하기 위해 만들었다. clk의 posedge에 동작할 수 있도록 always 문을 걸어준다. 그 후 reset\_n이 0일 경우 0을, set\_n이 0일 경우 1을, 이외의 경우에는 d를 출력하도록 한다.

* + 1. \_dff\_rs\_async

해당 모듈은 Asyncronous D Flip-Flop을 제작하기 위해 만들었다. clk의 posedge나 set\_n 또는 reset\_n이 negedge일 때 동작할 수 있도록 always 문을 걸어준다. 그 후 reset\_n이 0일 경우 0을, set\_n이 0일 경우 1을, 이외의 경우에는 d를 출력하도록 한다.

* + 1. \_dff\_rs\_sync\_async

해당 모듈은 Async와 Sync를 비교하기 위해 제작되었다. 해당 모듈을 각각의 q\_sync와 q\_async의 출력을 각각의 flip-flop에 연결하여 구성한다.

1. 설계 검증 및 실험 결과
   1. 시뮬레이션 결과

|  |
| --- |
|  |
| 해당 testbench는 \_dlatch를 검증한다. 처음에 clk이 0일 경우 d가 0이나 1이어도 초기 q가 없었기 때문에 x로 출력된다. 하지만 그 후 clk이 1일 경우 d값에 따라 값이 q로 출력되고 있다. 또한, 처음을 제외한 clk이 0인 경우에는 기존의 값을 그대로 유지하는 것 또한 볼 수 있다. |
|  |
| 해당 testbench는 \_dff를 검증하는 것이다. 해당 회로에서 초기에는 q에 기존의 값이 없었기 때문에 x가 출력된다. 하지만 그 후 clk이 뛸 때 뛸 당시에 d의 값에 따라 q가 출력된다. 이후에는 clk이 다시 뛰기 전까지 d값이 아무리 변하더라도 q는 변하지 않는 모습을 보인다. |
|  |
| 해당 testbench는 \_dff\_en을 검증하는 것이다. 해당 회로에서는 clk이 아무리 rising edge여도 en의 신호가 0이면 초기에는 x의 값(기존 q값), 후에는 기존의 q값을 출력하는 모습을 보인다. 그리고 en이 1일 경우는 \_dff의 회로를 사용한 것이기 때문에 간단하게 0과 1이 q에 들어가는지만을 확인하였다. 이미 앞에서 \_dff 회로를 검증했기 때문이다. |
|  |
| 해당 testbench에서는 \_dff\_r회로를 검증한다. reset을 1로 두었을 때에는 d의 값에 따라 q가 rising edge에서 변하는 모습을 보인다. 해당 부분은 이미 \_dff에서 검증했기 때문에 많은 경우를 실시하진 않았다. 만약 reset을 0으로 두었을 경우에는 다음 clk의 rising edge부터 d 값에 상관없이 0을 출력하는 모습을 보인다. |
|  |
| 해당 testbench에서는 \_dff\_rs 회로를 검증한다. reset과 set을 1로 두었을 때는 d의 값에 따라 정상적으로 값이 바뀌는 모습을 보인다. 해당 부분은 이미 \_dff에서 검증했기 때문에 많은 경우를 보이지 않았다. 만약 reset을 0로 두었을 경우에는 set의 값에 상관없이 0을 출력하는 모습을 보인다. 만약 reset은 1이고 set은 0인 경우 1을 출력하는 모습을 보인다. |
|  |
| 해당 testbench에서는 \_register32 회로를 검증한다. 해당 회로에 0000\_0000을 입력한 경우, 38af\_1297를 입력한 경우, cd0b\_564e를 입력한 경우 모두 데이터가 정상적으로 전달되는 모습을 보였다. |
|  |
| 해당 testbench에서는 \_dff\_rs\_async와 \_dff\_rs\_sync를 비교한다. async는 set나 reset이 0으로 떨어지는 즉시 값을 1 또는 0으로 초기화한다. 반면에 sync는 set이나 reset이 0으로 떨어지더라도 다음 rising edge에서 값이 바뀌는 모습을 보이고 있다. |

* 1. 합성(synthesis) 결과

|  |  |
| --- | --- |
| \_dlatch | |
|  |  |
| \_dff | |
|  |  |
| \_dff\_en | |
|  |  |
| \_dff\_r | |
|  |  |
| \_dff\_rs | |
|  |  |
| \_register32 | |
|  | |
|  |  |
| \_dff\_rs\_sync\_async | |
|  |  |
|  |  |

해당 모듈의 설계를 확인하면 각각의 모듈이 정상적으로 설계되어 연결되었음을 알 수 있다.

* 1. FPGA board targeting 결과

1. 고찰 및 결론
   1. 고찰

이번 주차의 과제를 진행하면서 처음에 D Latch가 동작하지 않아 어려움을 겪었다. 해당 문제는 필자가 always 문에 clk만을 넣어서 값이 제대로 전달되지 않았다. 이에 always@ (clk, n)과 같이 동시에 두 값을 넣어 클럭에 따라 값이 제대로 전달되도록 고쳤다. 위의 문제 이외에는 큰 어려움 없이 과제를 수행할 수 있었다.

* 1. 결론

해당 실험을 통해 Latch와 Flip-Flop이 어떻게 다른지 명확하게 이해할 수 있고, Flip-Flop을 가지고 여러가지 파생적인 회로를 만들 수 있다. 이를 통해 Register를 직접 구현할 수 있다. Syncronous와 Asynronous가 active low의 신호를 즉시 받는지, 또는 클럭이 올라갈 때 받는지에 따라 나눈다는 것을 알 수 있다.

1. 참고문헌

이준환 교수님/디지털논리회로1/광운대학교(컴퓨터정보공학부)/2021

이준환 교수님/디지털논리회로2/광운대학교(컴퓨터정보공학부)/2021