# 컴퓨터 공학 기초 실험2 보고서

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

실험일자: 2018년 09월 19 (수)

제출일자: 2018년 10월 02일 (화)


학 과: 컴퓨터공학과

담당교수: 공진흥 교수님

실습분반: 수요일 0, 1, 2

학 번: 2015722025

성 명: 정 용 훈

# 1. 제목 및 목적

## A. 제목

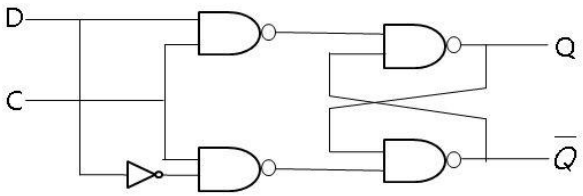
Latch & flip-flop design with/without reset/set

## B. 목적

여러 종류의 Latch와 Flip-Flop의 기능을 이해하며 이를 Verilog를 통하여 구현한다. 더 나아가 기본적인 Latch와 Flip-Flop을 응용한 Enabled Flip-Flop, Resettable Flip-Flop, Synchronous Set/Resettable Flip-Flop에 대하여 공부하고 Flip-Flop을 N개를 연결하는 N-bits register를 이해하며 구현한다.

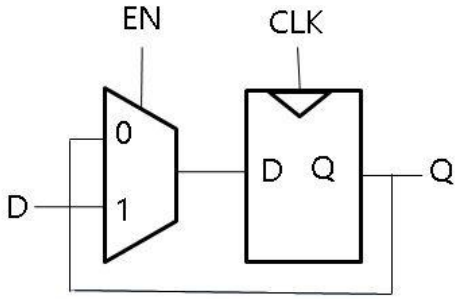
## 2. 원리(배경지식)

이번 실험은 1학기 과정에서 배운 기억 소자인 래치와 플립플롭을 다시하번 이해하며 베릴로그로 구현하는데 목적을 둔다. 기본적으로 래치와 플립플롭은 데이터를 기억하기 위한 소자이다. 각각의 소자를 구현하는 방법에는 여러가지의 게이트가 사용될 수 있다. 가장 쉽게 예를 들어보면 D latch를 구성하는데 있어 미리 제공된 강의 자료와 같이 구현할 수 있지만 아래 이미지 같이 Nand gate 4개를 사용하여 D latch를 구현할 수 있다.



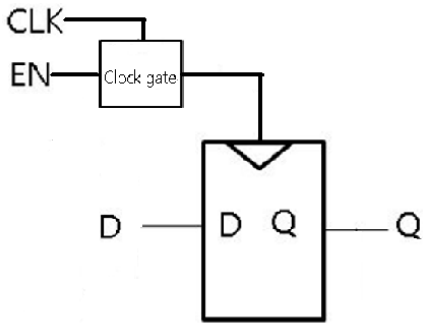

| Input | | Output | |
|---|---|---|---|
| **CLK** | **D** | **Q** | **Q'** |
| **0** | **X** | **$Q_{prev}$** | **$Q'_{prev}$** |
| **1** | **0** | **0** | **1** |
| **1** | **1** | **1** | **0** |

진리표를 비교하며 보았을 때 CLK의 값이 1로 rising 되어있을 때 비로소 소자가 정상적으로 동작을 하는 것을 볼 수 있다. CLK이 0인 상태에서는 D의 값에 관계없이 output이 계속 이전의 상태를 유지하는 것을 볼 수 있다. 비슷한 예로 enabled D Flop-Flop을 실습에서 구현하는 방법 말고 다른 방법으로 구현하는 방법을 알아보겠다. 우선 Enabled D Flip-Flop의 symbol과 진리표는 아래와 같이 볼 수 있다.



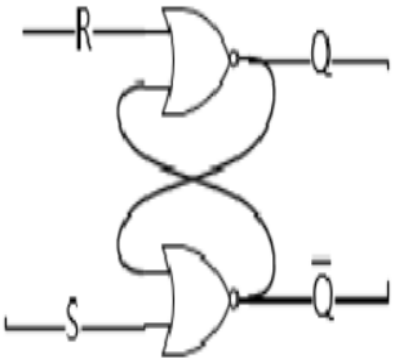

| Input | | | Output |
|---|---|---|---|
| **CLK** | **EN** | **D** | **Q** |
| **↑** | **1** | **1** | **1** |
| **↑** | **1** | **0** | **0** |
| **↑** | **0** | **X** | **$Q_{prev}$** |

동작 방법으로는 EN의 값이 1일 때는 기존의 D 플립플롭과 동작의 원리가 같다. 다른 점으로는 EN의 값이 0이 되면 mux의 output이 Q의 이전 값을 D에 전달 해주기 때문에 output이 이전의 값을 유지하며 D의 값을 바꾸어 주어도 output이 바뀌지 않는다. 위 Enabled D Flip-Flop을 실습에서 구현한 방법이 아닌 다른 방법으로 구현하면 아래와 같이 구현할 수 있다. 아래 그림은 mux를 사용하지 않고 CLK을 조절해주는 Clock gate를 사용하여 구현한 Enabled D Flip-Flop이다. 동작 방법으로는 EN의 값이 1이면 Clock이 정상적으로 동작하여 본래의 D 플립플롭의 동작을 하지만 EN의 값이 0이 되면 Clock이 동작하지 않아서 output의 값이 이전의 값을 계속 유지하는 결과가 나오게 된다. 이는 전에 mux와 설계한 D플립플롭과 같은 결과를 갖는다.



## 3. 설계 세부사항

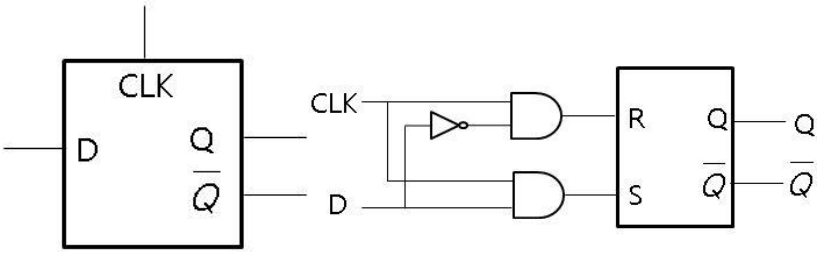
### a. SR Latch




| R | S | Qn | Qn+1 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | Qn | ?(부정) |

이번 설계를 위한 가장 기본적인 소자이다. 기억 소자로써 1-bits의 정보를 저장할 수 있는 소자이다. 옆 진리표는 각 인풋과 아웃풋의 상태에 대한 진리표이다. SR Latch는 앞으로 설계될 module들의 가장 기초가 되는 소자이므로 중요하다.

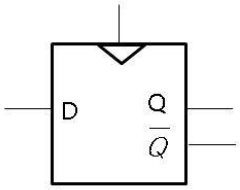
### b. D Latch




| **CLK** | **Q** |
|---|---|
| **0** | **$Q_{prev}$** |
| **1** | **D** |

위에 나와있는 symbol은 D latch의 symbol이다. 게이트로 나타내면 바로 옆에 나와있 이미지를 보면 된다. 인버터 하나와 and 두개와 SR latch로 이루어져 있으며 베릴로그로 구현하게 되면 SR latch를 instance 받아서 module을 구성하면 된다. 해당 module의 동작은 진리표를 확인하면 쉽게 비교할 수 있다.

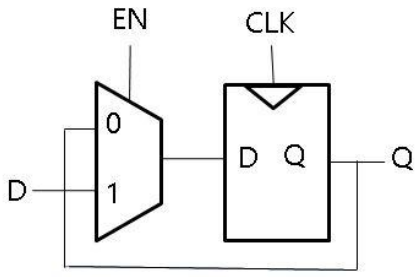
### c. D Flip-Flop




| **CLK** | **Q** |
|---|---|
| **↑** | **D** |
| **0 or 1 or ↓** | **$Q_{prev}$** |

위의 symbol은 D Flip Flop이다. 동작 자체는 D latch와 비슷하지만 clk의 edge에서만 값이 변한다는 차이가 있다. instance받아야 하는 module로는 D latch와 SR latch를 받아서 구현하면 된다. 해당 module의 동작은 진리표를 보면 비교할 수 있다.

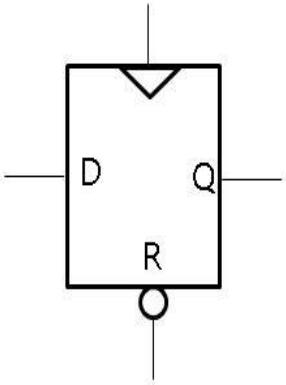
d. Enabled D Flip-Flop




| Input | | | Output |
|---|---|---|---|
| **CLK** | **EN** | **D** | **Q** |
| ↑ | **1** | **1** | **1** |
| ↑ | **1** | **0** | **0** |
| ↑ | **0** | **X** | $Q_{prev}$ |

앞서 원리에서 설명한 Enabled D Flip-Flop이다. En의 값에 따라 플립플롭의 동작을 기존의 D플립플롭처럼 동작하게 할 것인지 아니면 Q의 이전 값을 유지할 것인지 결정 할 수 있다. 해당 module은 이전의 D Flip Flop을 사용하며 2-to-1 mux를 설계하여 구현할 수 있다.

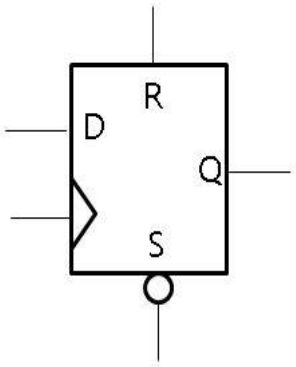
e. Resettable D Flip-Flop




| Input | | | Output |
|---|---|---|---|
| R | D | CLK | Q |
| 0 | X | X | 0 |
| 1 | 0 | ↑ | 0 |
| 1 | 1 | ↑ | 1 |
| 1 | X | 0, 1, falling edge | 이전 Q |

기존 D Flip Flop에 reset의 기능을 추가한 Flip Flop입니다. 하위 module로는 앞서 설계한 Enabled Flip Flop과 같습니다. Reset은 low active이며 뜻은 R의 값이 0이 되면 해당 기능을 동작하는 것입니다. 동작은 진리표에서 보는 것처럼 R이 0이면 output의 값이 0으로 고정되며 R이 1이면 기존의 D Flip Flop과 같은 동작을 합니다.

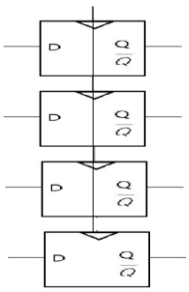
f. Synchronous Set/Resettable D Flip-Flop




| Input | | | | Output |
|---|---|---|---|---|
| R | S | D | CLK | Q |
| 0 | X | X | X | 0 |
| 1 | 0 | X | X | 1 |
| 1 | 1 | 0 | ↑ | 0 |
| 1 | 1 | 1 | ↑ | 1 |
| 1 | 1 | X | else | $Q_{prev}$ |

D Flip Flop에 set의 기능과 reset의 기능이 모두 포함된 Flip Flop이다. 두 기능 모두 low active이며 reset이 0이 되면 나머지 input의 값에 상관없이 output이 0이 되는 결과를 확인할 수 있다. 또한 reset이 1로 고정되어 있고 set이 0의 값을 갖는 다면 output이 1로 고정되는 것을 확인할 수 있다. 두 기능의 값 즉 R, S의 값이 1이 된다면 해당 module은 기존의 D Flip Flop의 기능을 한다. Synchronous는 동기식 방식으로 보고서 아래에 더 정확히 다루고 있다.

g. Register32



Register란 Flip Flop을 여러 개 연결하여 나열 한 module이다. 입력한 값을 이번 실습에서 설계한 rising edge에서 마다 output으로 나오는 결과를 확인 할 수 있다. 만들기 위해 instance되어야 할 module은 앞서 설계한 D Flip Flop을 포함한 Flip Flop을 설계하기 위한 module이 필요하다.

h. Async/Sync Set/Resettable D Flip-Flop

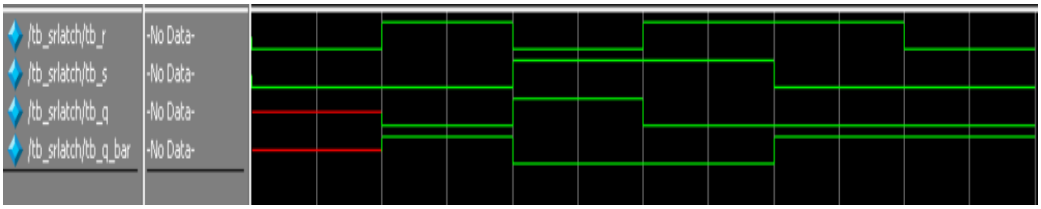
동기식 S/R D Flip Flop과 비동기식 S/R D Flip Flop은 symbol과 진리표자체는 일치한다고 봐도 무관하다. 동작으로는 차이가 있는데 동기식은set, reset 과 D의 값이 바뀌어도 그에 대한 신호처리가 clk의 edge에서만 바뀌는 특징이 있고 이에 반해 비동기식은 set과 reset의 처리가 clk의 dege에서만 바뀌는 것이 아니라 동작하는 순간에 바뀌게 된다. 즉 동기식은 신호의 대한 처리가 약속된 호출장소에서 동시에 일어나고 비동기식은 약속된 호출장소에서 동시에 일어나지 않아도 되는 것이다.

(Symbol의 내부 설계 사항은 합성결과에서 확인할 수 있다.)

## 4. 설계 검증 및 실험 결과

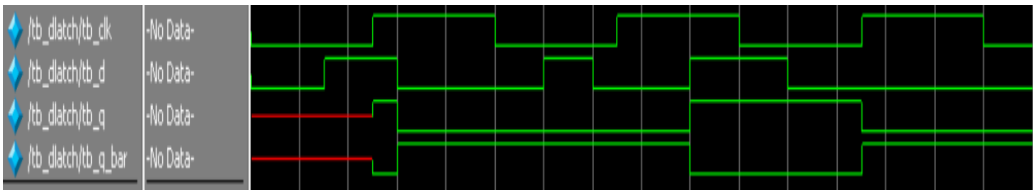
### A. 시뮬레이션 결과

#### SR Latch



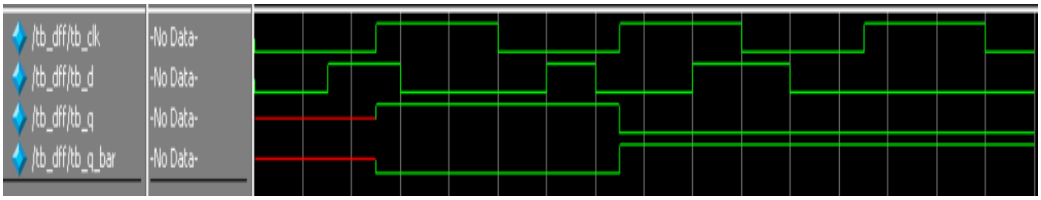

SR Latch의 경우 input의 값이 모두 0이면 이전의 output값을 유지하며 R의 값이 rising되면 reset으로 q의 값이 0이 된다 S의 값이 1로 rising되면 set으로 q의 값이 1이된다 한계점으로는 두개의 input이 모두 1의 값을 가지지 못한다.

#### D Latch



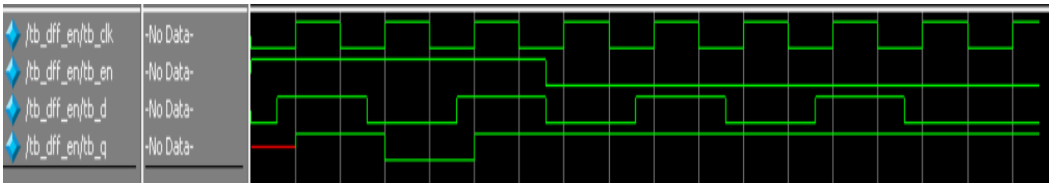

D Latch는 input으로 clk의 값과 data의 D 값이 들어가게 된다. 정상적인 동장은 clk가 rising 되어 있을 때 input D의 값에 영향을 받으며 clk가 0이면 D Latch는 이전의 값을 계속 유지하는 특징이 있다.

#### D Flip Flop



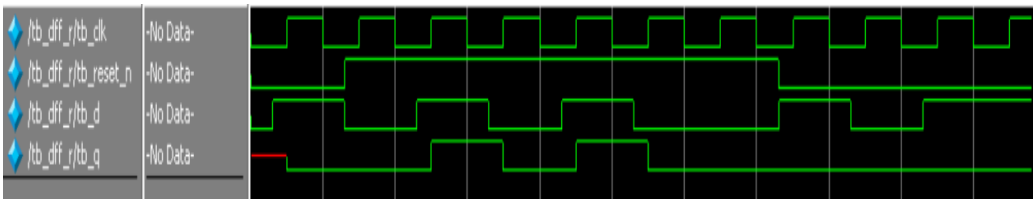

D Flip Flop과 Latch의 다른 점은 Flip Flop의 경우 clk의 rising하는 순간에 D의 값에 따라 output의 값이 결정되는 차이가 있다. Waveform을 보면 D의 값이 불규칙 하게 바뀌지만 output Q의 값은 rising하는 순간 D의 값에 따라 변하는 것을 볼 수 있다.

#### Enabled D Flip Flop



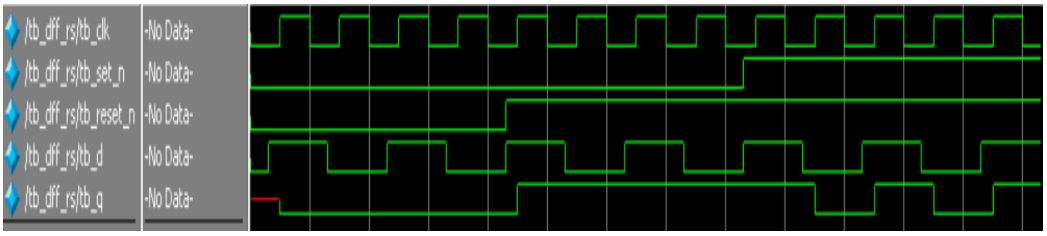

Enabled D Flip Flop은 en의 값에 따라 mux를 통하여 output의 값이 결정되는 Flip Flop이다. En의 값이 1로 rising되어있으면 기존의 D Flip Flop의 작동을 하지만 en의 값이 0이면 wave form에 나와있는 것처럼 q의 이전 값을 계속 유지하게 된다.

## Resettable D Flip Flop




이번 실습에서 구현된 resettable D Flip Flop은 active low에서 동작하는 방식이다. 좀 더 자세하게 예를 들어보면 reset의 값이 1이면 기존의 D Flip Flop기능을 하지만 reset의 값이 0을 유지하게 되면 clk에 따라 input의 값이 바뀌어도 wave form에 나와 있는 것처럼 q의 값이 이전의 값을 계속 유지하는 상태이다.

## Synchronous Set/Resettable D Flip-Flop

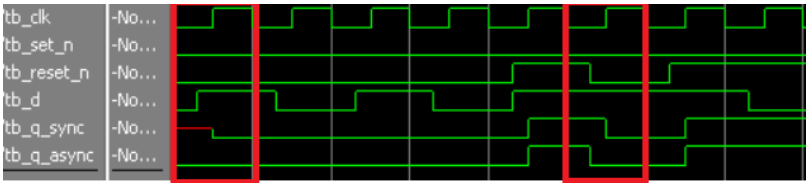



D Flip Flop기능에 set과 reset의 기능을 할 수 있는 Flip Flop이다 wave form을 살펴보면 두 동작이 low active이므로 reset이 0의 값을 가지고 있기 때문에 output의 값은 0으로 일정하게 유지되는 것을 볼 수 있다 다음 구간으로는 reset이 1로 rising되고 set이 0의 값을 유지하고 있기 때문에 output의 값이 1을 유지하고 있는 모습을 볼 수 있다. 다음 구간에서는 set와 reset 모두 1의 값을 가지고 있으므로 기본적인 D Flip Flop의 동작을 하는 것을 확인 할 수 있다.

## Register32


설계한 32비트 레지스터의 동작이다. 단순히 생각 하면 D Flip Flop을 bits의 수대로 연결하여 입력한 값이 있을 때 clock이 rising 하는 순간 입력한 값 그대로 output이 나오게 된다. Rising이 되지 않는 다면 아무리 input 값을 바꿔도 output의 값이 바뀌지 않는 특징이 있다.

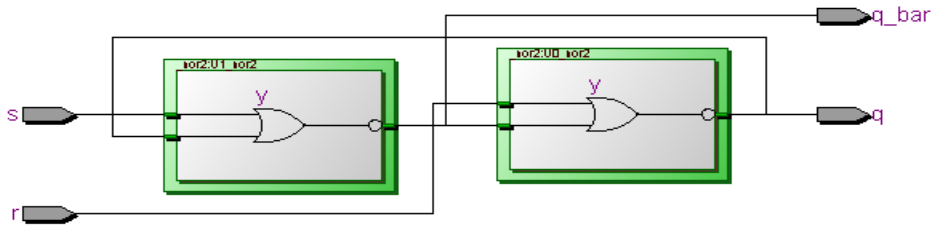
## Async/Sync Set/Resettable D Flip-Flop



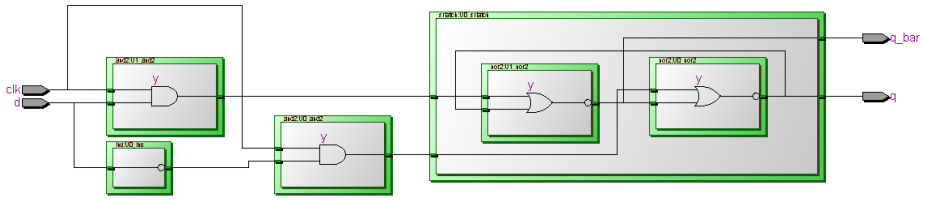

동작 자체는 같지만 신호가 처리되는 장소가 다르다. 동기식의 경우 클럭의 edge에서만 신호가 처리되고 비동기식인 경우에는 약속된 장소에 상관없이 set과 reset의 신호가 들어오면 바로 처리가 된다. 처음 네모의 경우 동기식은 신호가 처리되기 전이라서 Don't care이지만 비동기식은 reset과 set의 값을 처리하기 때문에 0으로 시작하는 것을 볼 수 있다.

B. 합성(synthesis) 결과

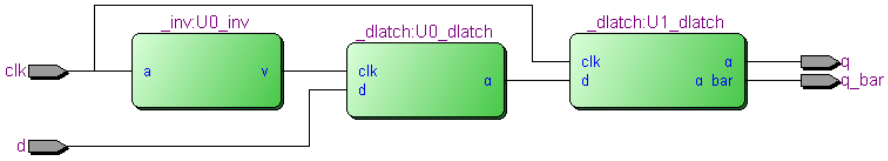
SR Latch



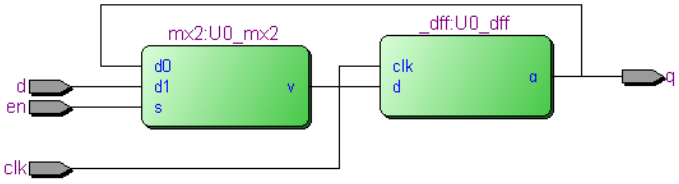

D Latch



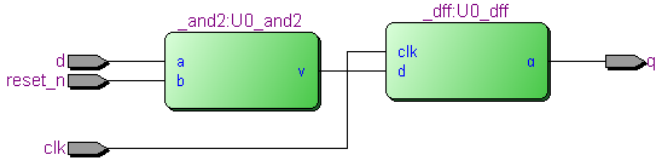

D Flip Flop



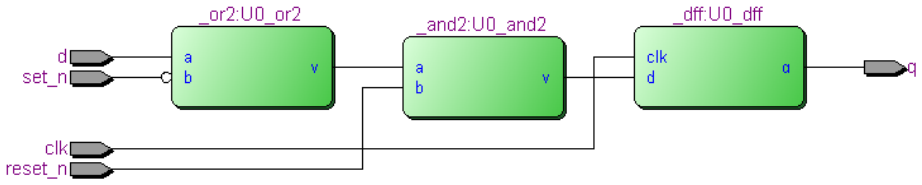

Enabled D Flip Flop



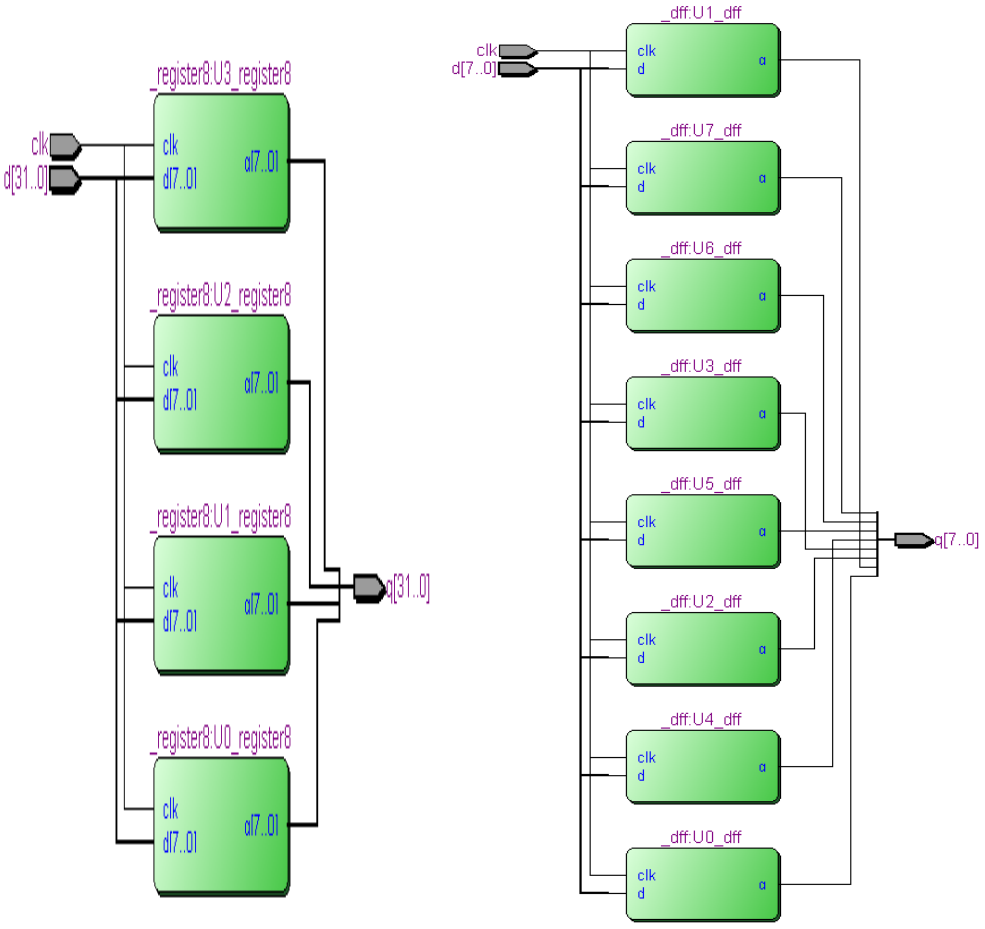

Resettable D Flip Flop



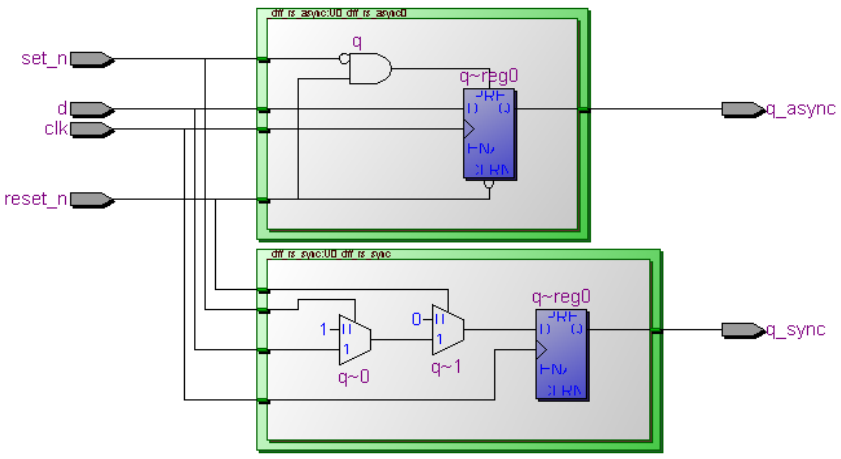
Synchronous Set/Resettable D Flip Flop




Register 32-bits




Async/Sync Set/Resettable D Flip-Flop



## 5. 고찰 및 결론

### A. 고찰

이번 과제에서는 미리 선수과목으로 배웠던 컴퓨터 공학기초 실험 1에서 중요하게 배웠던 기억소자인 latch와 여러 종류의 Flip Flop을 이해하고 구현하는데 목적을 두었다. 특히 이번 home work의 원리에서 Enabled D Flip Flop의 내용에서 mux를 대신하여 다른 방법으로 Flip Flop을 설계하는데 고민을 많이 하였다. 해결 방법부터 얘기하자면 다른 input을 기준으로 설계하는 것을 생각하는 게 아니라 Clock자체를 멈추는 방법을 생각할 수 있는 접근법, Clock gate를 사용하여 일정 값이 들어가면 Clock을 멈추어 값을 변환시키지 않을 수 있도록 설계하는 것이 방법이었다. (Clock gate 대신 And gate를 사용할 수 있다.) 하지만 처음에는 Clock을 생각하지 않고 단순히 input값과 그에 대한 gate들을 사용하여 값을 도출하려고 하니 생각처럼 쉽지 않았다. 검색엔진을 통하여 Enabled Flip Flop을 검색하게 되었고 Flip Flop을 설계하는 다른 방법으로 Clock을 멈추는 방법을 알게 되었다. 그리하여 Clock을 멈추는 장치인 Clock gate를 사용하여 다른 방법으로 Enabled Flip Flop을 설계하게 되었다. Clock gate 대신 and gate를 사용하여 clock을 멈출 수 있는 방법도 있다.

### B. 결론

해당 실습을 진행하면서 동기식과 비동기식의 대한 개념을 좀 더 알아갈 수 있는 계기가 되었습니다. 우선 동기식과 비동기식의 차이점으로는 쉽게 말해 요청에 따른 응답이 동시에 일어나면 동기식이고, 작업의 처리가 동시에 일어나지 않으면 비동기식이라고 할 수 있다. 쉽게 예를 들어보면 동기식 Flip Flop은 set의 값과 reset의 값 D의 값이 clk의 edge에서만 처리가 되고 비동기식 Flip Flop은 clk에서는 D의 값만 처리해주고 set과 reset의 대한 명령은 따로 실행이 되는 것이다. 또한 동기식 과 비동기식의 방식은 통신에서도 많이 쓰이게 된다. 두 방법모두 장단점이 있기 때문에 이 문제를 응용해서 사용하기 위해서는 적절히 전송 방법에 대하여 이해하고 있어야 한다.

## 6. 참고문헌

Clock gating / https://en.wikipedia.org/wiki/Clock_gating

김영민/Overview of Verilog - Sequential Circuits/광운대학교/2018-2

이준환/Latch & Flip Flop/광운대학교/2018-1