컴퓨터 공학 기초 실험 2 – Lab 4

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

# D Latch

이번 실습에서는 이전에 값을 유지하고 있는 저장 소자 역할을 하는 latch와 flip-flop을 설계하고, 더불어 reset과 set 기능을 구현한다. 또한, 구현한 flip-flop을 사용하여 N-bits register를 구현하는 방법에 대하여 살펴보도록 한다

* Description
* D latch는 clock이 enable 상태를 유지하는 동안 입력 D 값의 변화를 출력한다.
* 다음은 D latch의 symbol과 truth table이다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| EMB00001a24824d  Figure 1 - D latch의 symbol | |  |  | | --- | --- | | CLK | Q | | 0 | 이전 Q | | 1 | D |   Table 1- D latch의 truth table |

* Structural Description

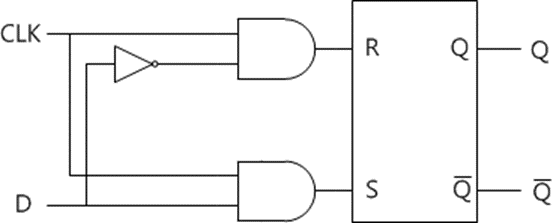


Figure 2 . D-latch Schematic

# D Flip-Flop

* Description
* D flip-flop은 clock의 rising edge나 falling edge에서만 D 값으로 출력이 바뀌게 된다. 다른 경우에는 D 값이 바뀌더라도 이전 Q 값을 그대로 유지한다. 해당 실습에서는 clock의 rising edge를 사용한다.
* 다음은 D flip-flop의 symbol과 truth table이다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| EMB00001a248250  Figure 3 - D flip-flop의 symbol | |  |  | | --- | --- | | CLK | Q | | ↑ | D | | in other case | 이전 Q |   Table 2 - D flip-flop의 truth table |

* Structural Description

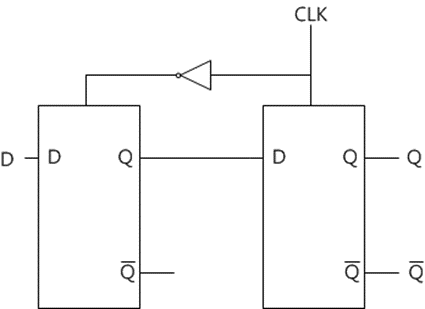


Figure 4 . D-latch Schematic

# Comparison of D Latch and D Flip-Flop

* 다음 그림을 통해 D latch와 D flip-flop의 차이를 알 수 있다.

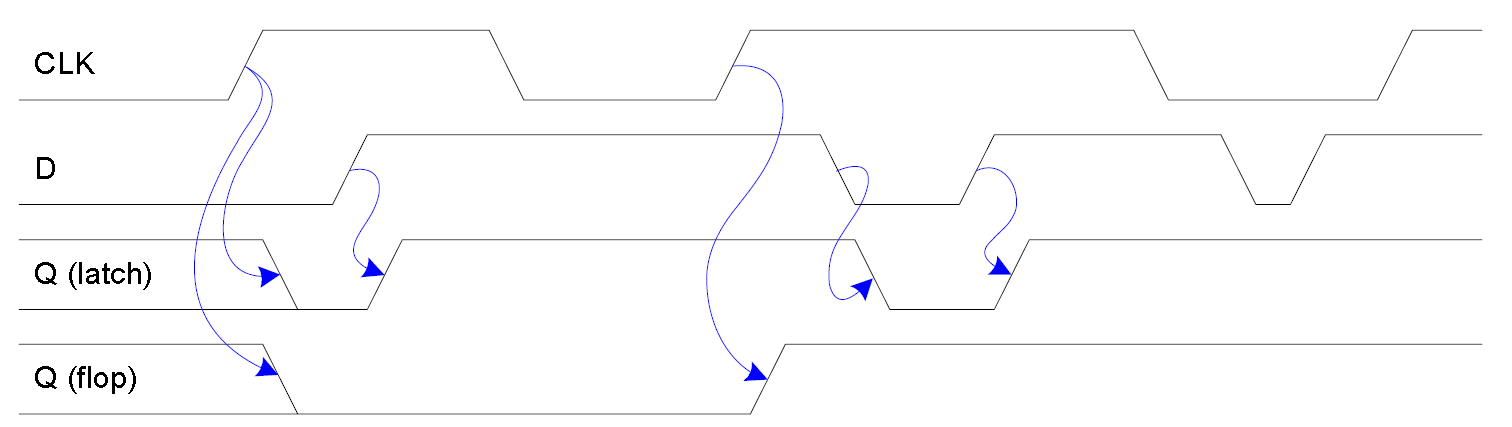


Figure 5 - Waveform of D Latch and D Flip-Flop

# Resettable D Flip-Flop

* Description
* Resettable D flip-flop은 D flip-flop에 reset 기능이 추가된 D flip-flop이다. 실습에서 구현하는 resettable D flip-flop은 active low에 동작한다. Active low에 동작한다는 reset의 값이 0일 때, reset 기능을 수행한다는 의미이다.
* 다음은 resettable D flip-flop의 symbol과 truth table이다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Figure 6 – Resettable D FF의 symbol | |  |  |  |  | | --- | --- | --- | --- | | Input | | | Output | | R | D | CLK | Q | | 0 | X | X | 0 | | 1 | 0 | ↑ | 0 | | 1 | 1 | ↑ | 1 | | 1 | X | ↓ or 0 or 1 | 이전 Q |   Table 3 - Resettable D FF의 truth table |

# 실습 (실습 강의자료를 함께 참고할 것)

* D Latch
* Implementation  
  앞서 설명한 D latch의 schematic은 다음 그림과 같다.

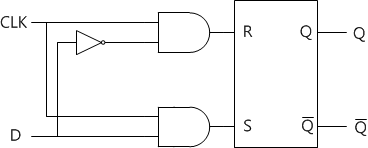


Figure 10 - D latch의 schematic

앞서 구현한 SR latch의 입력에 and gate와 not gate를 사용하여 D latch를 구조적으로 구현할 수 있다.

\_dlatch라는 이름으로 project와 top module을 만들고, D latch를 구현 후 검증하여 본다.

* Top module의 이름을 dlatch는 경우, verilog에서 사전에 정의된 primitive이기 때문에 사용할 수 없다.
* D Flip-Flop
* Implementation  
  앞서 설명한 D flip-flop은 D latch 2개를 이용하여 구조적으로 구현할 수 있다. 다음 그림은 D flip-flop의 schematic이다.

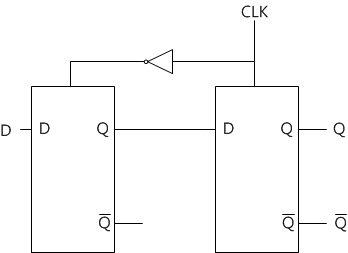


Figure 11 - D flip-flop의 schematic

\_dff라는 이름으로 project와 top module을 만들고, D flip-flop을 구현 후 검증한다.

* Enabled D Flip-Flop
* Implementation  
  Enabled D FF는 D FF 앞에 multiplexer를 추가하여 입력 data를 사용할 지를 결정한다. 다음 그림은 enabled D FF의 schematic이다.

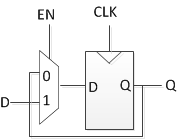


Figure 12 – Enabled D flip-flop의 schematic

\_dff\_en라는 이름으로 project와 top module을 만들고, enable D FF를 구현 후 검증한다.

* Resettable D Flip-Flop
* Implementation  
  Resettable D FF는 D FF 앞에 and gate를 추가하여 입력 data를 초기화할 지를 결정한다. 다음 그림은 resettable D FF의 schematic이다.

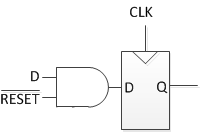


Figure 13 – Resettable D flip-flop의 schematic

\_dff\_r라는 이름으로 project와 top module을 만들고, resettable D FF를 구현 후 검증한다.

* Synchronous Set/Resettable D Flip-Flop
* Implementation  
  해당 실습에서는 reset과 set 기능을 가지는 D flip-flop을 구현하여 본다. 여기서 active low의 의미는 reset이나 set이 각각 0이 되었을 때, reset 또는 set의 기능을 함을 의미하며, synchronous는 clock에 동기화가 맞춰지는 것을 의미한다. 이를 구조적으로 구현하기 위하여 앞서 구현한 D flip-flop과 and, or, not gate를 이용한다. 다음 그림은 이에 대한 schematic이다.

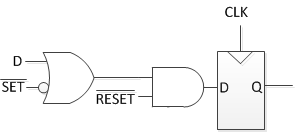


Figure 14 – Set/resettable D flip-flop의 schematic

\_dff\_rs라는 이름으로 project와 top module을 만들고, set/resettable D FF를 구현 후 검증한다.

* Register
* Description  
  N-bits register는 N 개의 flip-flop을 한 줄로 늘여놓음으로써 구현한다. 다음 그림은 32 bits register의 symbol을 나타낸다.

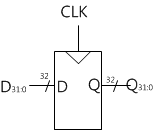


Figure 15 – 32bits register의 symbol

* Implementation  
  다음 그림은 32 bits register의 schematic이다.

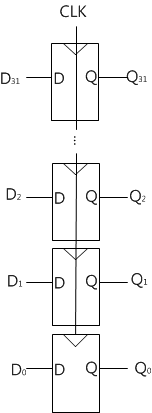


Figure 16 - 32bit register의 schematic

(32-bit register를 구현 시 D FF를 32개 instantiation하여도 되고, 8-bit register를   
구현 후 이를 4개 instantiation하여 32-bit register를 구현하여도 무관하다.  
하지만, 이번 실습에서는8-bit register를 instance사용하는 것을 기준으로 한다.)

* Async/Sync Set/Resettable D Flip-Flop
* 앞선 flip-flop(또는 register)의 구현은 structural implementation을 하였다. 이번에는 behavioral implementation을 통하여 asynchronous/synchronous의 차이를 확인하는 것이 목적이다.
* Implementation of Synchronous Set/Resettable D Flip-Flop

다음 그림은 synchronous set/resettable D FF의 code이다.

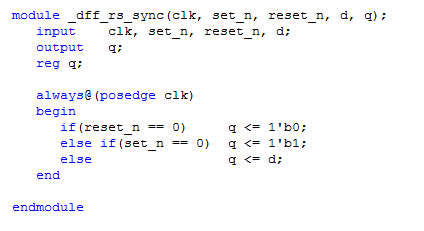


Figure 17 – Implementation of synchronous set/resettable D FF

* Implementation of Asynchronous Set/Resettable D Flip-Flop

다음 그림은 asynchronous set/resettable D FF의 code이다.

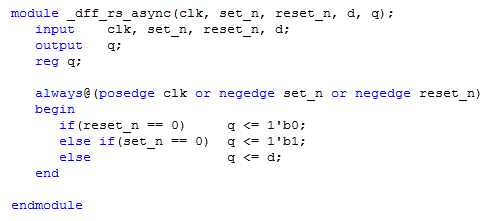


Figure 18– Implementation of asynchronous set/resettable D FF

* Implementation of Top Module

다음 그림은 앞선 두 module을 instance한 top module이다.

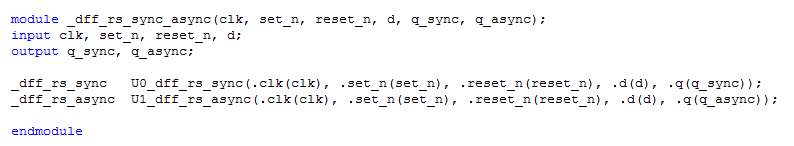


Figure 19 - Implementation of top module

* Top module에 대한 testbench를 작성하여 두 개의 차이점을 확인하여 본다.

(Hint : set\_n과 reset\_n이 모두 active low에 동작한다.)

# Report

레포트는 공지사항에 올린 보고서 양식에 맞추어 작성하고, 다음의 사항에 대하여서도 추가적으로 작성한다.

* D latch, D flip-flop, enabled D flip-flop, resettable D flip-flop, 32 bits register, async/sync set/resettable D flip-flop에 대하여 구현하고 검증한다.
* RTL viewer는 포함한다.
* 원리(배경지식)에 nand gate 4개를 이용하여 D latch을 구현하는 방법에 대하여 조사하고 설명한다.
* 원리(배경지식)에 실습에서 구현한 enabled D flip-flop이 아닌 다른 방법으로 enabled D flip-flop을 구현하는 방법에 대하여 조사하고 설명한다.
* 고찰 및 결론에 D FF with active-low synchronous reset and set과 D FF with active-low asynchronous reset and set의 차이를 설명하여라.

# Submission

* 제출할 프로젝트
* 해당 실습은 프로젝트 폴더가 아닌 **Verilog file만을 하나의 폴더에 모아서 제출**한다. 제출할 file list는 다음과 같다. 그 외에 필요하다고 생각되는 파일을 추가하여도 된다.  
  **(지켜지지 않을 경우 감점)**
* Module file  
  gates.v, mx2.v, \_dlatch.v, \_dff.v, \_dff\_en.v, \_dff\_r.v, \_dff\_rs.v, \_register8.v, \_register32.v, \_dff\_rs\_sync.v, \_dff\_rs\_async.v, \_dff\_rs\_sync\_async.v
* Testbench file  
  tb\_dlatch.v, tb\_dff.v, tb\_dff\_en.v, tb\_dff\_r.v,, tb\_register32.v,  
  tb\_dff\_ rs\_sync\_async.v
* 월요일 반: 10월 19일 23시 59분까지
* 금요일 오전반: 10월 19일 23시 59분까지
* 금요일 오후반: 10월 19일 23시 59분까지
* **Source code와 report를 같이 ZIP 파일로 압축하여 KLAS과제 제출에 해당 과제 upload**
* 업로드 파일명은 (요일#)\_(학번)\_Assignment\_#.zip
* 요일번호는 실습 미수강은 0, 월요일 0, 1, 2교시은 1, 금요일 0, 1, 2 교시는 2, 금요일 5, 6, 7 교시는 3
* Ex) 월요일 반 수강, 2019110609, Assignment 1 제출 시

1\_2019110609\_Assignment\_1.zip으로 제출

* Report 명은 (요일#)\_(학번)\_(이름)\_Assignment\_#.pdf
* 요일 번호는 위의 업로드 파일명과 동일하게 진행
* Ex) 월요일 반 수강, 2019110609, 정현우, Assignment 1 제출 시

1\_2019110609\_정현우\_Assignment\_1.pdf으로 제출

* Report는 PDF로 변환해 제출 (미수행시 감점)
* Source code 압축 시 db, incremental\_db, simulation 폴더는 삭제(미수행시 감점)
* 실습자료와 다르게 instance를 요구한 모듈에서 behavioral하게 구현한 경우 (gate혹은 모듈을 instance하지 않을 시) 감점
* 채점기준

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 세부사항 | | 점수 | 최상 | 상 | 중 | 하 | 최하 |
| 소스코드 | Source code가 잘 작성 되었는가?  (Structural design으로 작성되었는가?) | 10 | 10 | 8 | 5 | 3 | 0 |
| 주석을 적절히 달았는가?  (반드시 영어로 주석 작성) | 20 | 20 | 15 | 10 | 5 | 0 |
| 설계검증  (보고서) | 보고서를 성실히 작성하였는가?  (보고서 형식에 맞추어 작성) | 30 | 30 | 20 | 10 | 5 | 0 |
| 합성결과를 설명하였는가? | 10 | 10 | 8 | 5 | 3 | 0 |
| 검증을 제대로 수행하였는가?  (모든 입력 조합, waveform 설명) | 30 | 30 | 20 | 10 | 5 | 0 |
| 총점 | | 100 |  | | | | |