10주차 결과보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20191599 이름: 송경호

**1.**

RS Flip-Flop은 두 개의 입력을 바탕으로 두 개의 출력을 발생시키는 회로이다. 두 개의 입력은R과 S로, 각 각 Reset과 Set을 의미하며, 해당 입력에 따라 출력 Q와 Q\*이 변화한다. 다음은 RS Flip-Flop의 진리표이다.

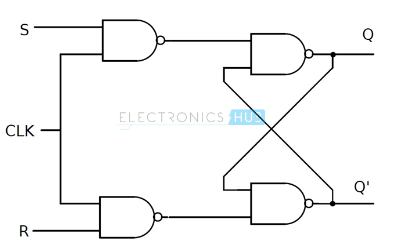
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | **Output** | |
| **Clk** | **R** | **S** | **Q** | **Qp** |
| 0 | x | x | Q | Qp |
| 1 | 0 | 0 | Q | Qp |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | x | x |

진리표를 확인하면 Clock입력을 의미하는 CLK bit가 추가적으로 존재한다. Clk 입력이 0일 때는 R과 S의 입력이 무시되어 출력은 이전의 출력을 이어 가지며, Clk 입력이 1일 때만 R과 S에 따른 Q와 Qp의 변화가 발생한다. 먼저 R과 S가 각 각 (0,0)일 때는 Clk입력이 0일 때와 마찬 가지로, 이전의 출력을 그대로 갖는다. 다음으로 (0,1)의 입력을 갖을 때, 다시 말해 Set bit만이 1일 때 Q는 1을 Qp는 0을 갖는다. 다음으로 Reset bit 만이 1인 (1,0)의 입력에선 이전과 완전히 반대로 Q는 0을 Qp는 1을 갖게 된다. 마1-1지막으로, RS Flip-Flop은 (1,1)의 입력을 허용하지 않으므로 위와 같은 진리표가 생성된다.

RS Flip-Flop은 두 가지 형태로 나타낼 수 있다.

**1-1) NAND RS Flip-Flop**

NAND RS Flip-Flop의 회로는 다음과 같다.



NAND RS Flip-Flop은 4개의 NAND로 이루어져 있다. 해당 회로를 Verilog를 통해 구현한 결과는 아래와 같다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

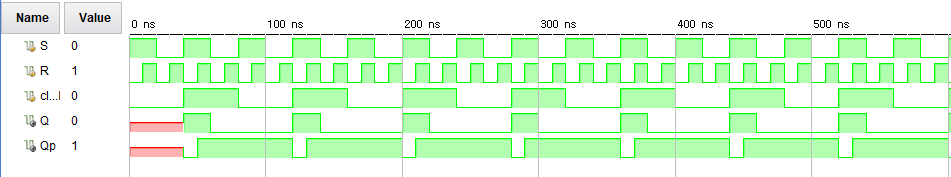
자동 생성된 설명텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

두 출력은 R과 S중 하나의 입력과 clock을 NAND 연산한 결과와 나머지 하나의 출력을 NAND한 결과이다. 따라서 중간 과정에서 계산되어야 하는 하나의 입력과 NAND의 연산 결과를 wire t1, t2에 담아 보다 직관적일 수 있도록 디자인 코드를 구현했다. 추가적으로 해당 실험에서는 편의를 위해 엣지 트리거를 구현하지 않고 단순히 clk 비트를 토글하는 방식으로 구현했다.

테스트 밴치 코드를 보면 R과 S는 각 각 10, 20ns마다 전환되며 clock 비트는 40ns마다 전환되는데 이 때, clock 비트가 올라와 있는 상태에서는 flip-flop이 아닌 latch처럼 동작하는 것을 알 수 있다. 시뮬레이션 결과를 보다 쉽게 보기 위하여, clock 비트의 변환 시간을 가장 길게 하였다.

다음은 해당 코드의 시뮬레이션 결과이다.



위에서부터 S, R, clock 비트가 각 각 20, 10, 40ns 마다 전환되는 것을 알 수 있다. 이때, 40ns에 처음으로 clock 비트가 1이 되고 S와 R이 각 각 (1,0)이므로 Q와 Qp가 각 각 (1,0)이 된다. 또한 70ns~80ns 지점에서 S와 R이 각 각 (0,1)이므로 Q와 Qp가 각 각 (0,1)이 된다. 다음으로 80ns 지점에서 clock이 다시 0으로 전환되며 이전의 출력 결과가 이어지는 것을 알 수 있다. 마지막으로 50ns~60ns 지점에서 (1,1)의 입력과 다음 60ns~70ns 지점에서의 (0,0)의 입력이 진리표와 다르게 입력이 (0,0)임에도 불구하고 출력이 변하는 것을 알 수 있는데, 이는 이전 값 (1,1)로 인해 허용되지 않은 출력이 발생했기 때문이다.

(0,0) 입력에서 출력이 변하지 않는 결과를 확실히 확인하고자 개인 PC에서 test bench코드를 바꾸어 시뮬레이션을 다시 진행했다.

스크린샷, 사각형, 디자인이(가) 표시된 사진

자동 생성된 설명

이전 실험의 문제는 (1,1)의 입력 이후 바로 (0,0)의 입력이 나오는 것이었기 때문에, 테스트 밴치 코드의 주기만을 변경하였다. 결과론적으로 530ns에서 S,R이 (0,1)인 입력에 의해 Q와 Qp가 (0,1)로 발생된 이후로 (0,0)의 입력에서 해당 출력이 계속해서 이어진 것을 확인할 수 있다.

**1-2) NOR RS Flip-Flop**

NOR RS Flip-Flop의 회로는 다음과 같다.

도표, 라인, 폰트, 그래프이(가) 표시된 사진

자동 생성된 설명

NOR RS Flip-Flop은 이전과 달리 NAND 대신 AND와 NOR 회로로 이루어져 있다. 또한 주의해야 할 점은 R과 S의 입력이 Q와 Qp에 NAND Flip-Flop의 반대로 입력된다는 점이다. 이 점을 주의하며 해당 회로를 Verilog를 통해 구현한 결과는 아래와 같다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

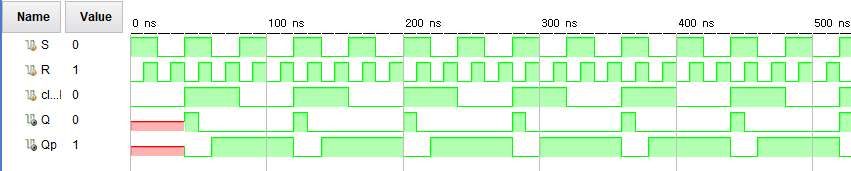
자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

두 출력은 R과 S중 하나의 입력과 clock을 AND 연산한 결과와 나머지 하나의 출력을 NOR한 결과이다. NAND Flip-Flop과 마찬가지로 중간 과정에서 계산되어야 하는 입력을 wire를 이용해 보다 직관적일 수 있도록 디자인 코드를 구현했다.

테스트 밴치 코드는 NAND Flip-Flop과 완전히 동일하다. R과 S는 각 각 10, 20ns마다 전환되며 clock 비트는 40ns마다 전환되는데 마찬가지로 clock 비트가 올라와 있는 상태에서는 flip-flop이 아닌 latch처럼 동작하는 것을 알 수 있다. 또한 동일하게 시뮬레이션 결과를 보다 쉽게 보기 위하여, clock 비트의 변환 시간을 가장 길게 하였다.

다음은 해당 코드의 시뮬레이션 결과이다.



우선 NAND Flip-Flop과의 차이점은 허용되지 않은 입력인 (1,1)의 입력에서 NAND는 (1,1)이 출력되는 반면 NOR은 (0,0)이 출력된다는 점이다. 이는 최종 출력이 각 각 NAND Gate와 NOR Gate이기 때문에 발생하는 것으로 두 회로 모두 하나의 출력이 나머지 출력에 관여하기 때문에 이러한 결과가 발생한다.

위에서부터 S, R, clock 비트가 각 각 20, 10, 40ns 마다 전환되는 것을 알 수 있다. NAND Flip-Flop과 동일하지만 다시 한번 설명하자면 40ns에 처음으로 clock 비트가 1이 되고 S와 R이 각 각 (1,0)이므로 Q와 Qp가 각 각 (1,0)이 된다. 또한 70ns~80ns 지점에서 S와 R이 각 각 (0,1)이므로 Q와 Qp가 각 각 (0,1)이 된다. 다음으로 80ns 지점에서 clock이 다시 0으로 전환되며 이전의 출력 결과가 이어지는 것을 알 수 있다. 마지막으로 50ns~60ns 지점에서 (1,1)의 입력과 다음 60ns~70ns 지점에서의 (0,0)의 입력이 진리표와 다르게 입력이 (0,0)임에도 불구하고 출력이 변하는 것을 알 수 있는데, 이는 이전 값 (1,1)로 인해 허용되지 않은 출력이 발생했기 때문이다.

(0,0) 입력에서 출력이 변하지 않는 결과를 확실히 확인하고자 다시한번 개인 PC에서 test bench코드를 바꾸어 시뮬레이션을 다시 진행했고 아래와 같이, 530ns 지점에서 S, R이 (0,1)인 입력에 의해 Q와 Qp가 (0,1)로 발생된 이후로 (0,0)의 입력에서 해당 출력이 계속해서 이어진 것을 확인할 수 있다.

스크린샷, 텍스트, 사각형이(가) 표시된 사진

자동 생성된 설명

**2.**

D Flip-Flop은 RS Flip-Flop과 거의 동일한 형태를 띄고 있지만, 입력이 R과 S, 두 개가 아닌, D만 존재한다는 것이 가장 큰 차이점이다. 그럼에도 불구하고, RS Flip-Flop과 형태가 비슷한 점은 R과 S의 자리가 단순히 D와 ~D로 대체되기 때문이다. 다음은 D Flip-Flop의 진리표이다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
| **Clk** | **D** | **Q** | **Qp** |
| 0 | x | Q | Qp |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

D Flip-Flop역시 진리표에 보이듯 Clock입력이 존재하며 역시 마찬가지로 Clk 입력이 0일 때는 D입력이 무시되어 출력은 이전의 출력을 이어 가지며, Clk 입력이 1일 때만 D에 따른 Q와 Qp의 변화가 발생한다. 먼저 이전과 달리 입력이 D와 ~D이므로 RS 기준 (0,0)와 (1,1)의 입력은 고려하지 않는다. 대신 D가 0인 경우 RS 기준 (0,1)의 입력을 갖게 되어, Q는 1을 Qp는 0을 갖고 D가 1인 경우 RS 기준 (1,0)의 입력을 갖게되어 Q는 0을 Qp는 1을 갖게 된다.

D Flip-Flop의 회로는 다음과 같다.

도표, 라인, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

NOR RS Flip-Flop의 구조를 가지며 앞서 설명한대로 RS 대신 D와 ~D가 입력으로 작용한다. 이를 바탕으로 Verilog 코드를 구현한 결과는 아래와 같다.

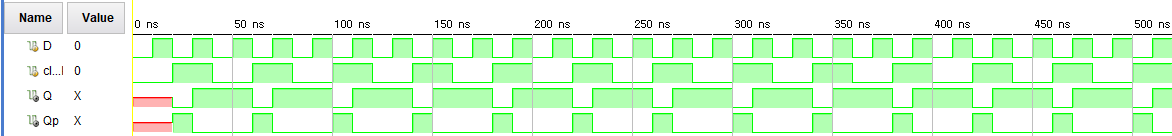
텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷, 대수학이(가) 표시된 사진

자동 생성된 설명

위의 회로 그림는 NOR RS Flip-Flop 형태의 D Flip-Flop이지만, Verilog는 NAND RS Flip-Flop 형식을 구현하였다. NAND RS Flip-Flop에서 설명했듯이, wire를 이용하여 중간 계산 결과를 저장했다. 이외의 구성 모두 R S의 입력을 D와 ~D로 변환한 것을 제외하고 NAND RS Flip-Flop과 동일하다.

다음은 해당 코드의 시뮬레이션 결과이다.



20ns에 처음으로 clock 비트가 0이 되고 D가 0이므로 RS Flip-Flop에서 S와 R이 (0,1)인 것과 동일한 작용을 하여 Q와 Qp가 각 각 (0,1)이 된다. 다음으로 30ns~40ns 지점에서 D가 1이 되어 RS Flip-Flop에서 S와 R이 (1,0)인 것과 동일한 작용을 하여 Q와 Qp가 각 각 (1,0)이 된다. 이후로 다시 clock비트가 0으로 전환되어 이전의 출력이 유지되며 해당 과정을 반복한다.

**3.**

Flip-Flop의 동작을 이해하고 회로를 구성하여 해당 회로를 Verilog로 구현하는 과정을 RS Flip-Flop과 D Filp-Flop에 대하여 수행했다. 또한 해당 구현의 시뮬레이션과 진리표가 동일한 결과를 생성하는 것을 확인하였다.

또한 해당 실험에서는 edge trigger에 관한 코드를 따로 작성하지 않고 clock 비트를 단순히 토글하는 방식으로 구현했으며 Clear bit 역시 고려하지 않아 비교적 간단한 실험이 수행되었다.

또한 RS Flip-Flop에서 FPGA의 실행 과정에서는 아무런 오류가 발생하지 않았지만 결과를 분석하는 데에 있어 (1,1)의 입력 이후로 곧바로 (0,0)의 입력이 발생하도록 테스트 밴치 코드를 작성하여 시뮬레이션 결과 분석에 어려움이 있었다. 해당 문제는 테스트 밴치 코드를 수정하여 해결했다.

마지막으로 Flip-Flop이 아닌 Latch에 대한 실습 내용도 언급되어 있는데 이는 단순히 위의 실습에서 Clock 비트를 제거하여 구현 가능하다.

**4.**

실험을 수행하지 않았지만 추가적인 Flip-Flop들이 존재한다.

**4-1) JK Flip-Flop**

JK Flip-Flop은 RS Flip-Flop이 (1,1)의 입력을 허용하지 않는 단점을 보완한 Flip-Flop으로 (1,1)의 입력 시 현재 Q를 유지하고 해당 Q의 부정을 취한 값을 Qp로 출력한다. JK Flip-Flop의 진리표와 회로는 아래와 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| **J** | **K** | **Q (Prev)** | **Q\* (Next)** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

도표, 스케치, 라인, 기술 도면이(가) 표시된 사진

자동 생성된 설명

해당 회로 역시 Verilog 코드로 구현할 수 있다. NOR RS Flip-Flop에서 출력 Q와 Qp를 연산의 중간 단계인 wire t1과 t2를 구하는 연산의 입력으로 넣어 구현 가능하다.

**4-2) T Flip-Flop**

RS Flip-Flop과 D Flip-Flop의 관계처럼 JK Flip-Flop 역시 입력을 하나로 줄인 T Flip-Flop이 존재한다. T Flip-flop은 T의 값이 0일 때는 Q의 값이 그대로 Qp로 전달되며, T의 값이 1일 때는 Q의 값에 보수 형태가 Qp로 전달된다. 아래는 T Flip-Flop의 진리표와 회로도이다.

|  |  |  |
| --- | --- | --- |
| **T** | **Q (Prev)** | **Q\* (Next)** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

도표, 라인, 스케치, 기술 도면이(가) 표시된 사진

자동 생성된 설명

해당 회로 역시, JK Flip-Flop과 마찬가지로 Verilog 코드 구현이 가능하다.