**11주차 결과보고서**

**전공: 경영학과 학년: 3학년 학번: 20190963 이름: 한다현**

**1.**

**RS 플립플롭은 두 입력 R과 S로 두 출력 Q와 Q’을 결정하는 플립플롭이다. 이 때 R은 reset을, S는 set을 의미하며 R이 1이 될 때 Q는 0이 되고 Q’은 1이 되며 S가 1일 때 Q는 1이 되고 Q’은 0이 된다. 또한 두 입력 R, S 외에도 clock pulse가 입력되어야 하는데, clock pulse가 1인 경우에만 플립플롭이 작동되어야 하고, 0인 경우에는 작동하지 않는다. RS 플립플롭의 behavioral table은 다음과 같다.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CLK** | **R** | **S** | **Q** | **Q’** |
| **0** | **X** | **X** | **Q** | **Q’** |
| **1** | **0** | **0** | **Q** | **Q’** |
| **1** | **0** | **1** | **1** | **0** |
| **1** | **1** | **0** | **0** | **1** |
| **1** | **1** | **1** | **Not allowed** | |

**CLK가 0일 때는 R, S 값과 상관없이 플립플롭이 작동되지 않으므로 Q와 Q’의 값은 이전과 동일하다. CLK가 1인 경우에, R의 값이 1이면 Q값은 0이 되고, S의 값이 1이면 Q 값은 1이 된다. 또한 R과 S가 모두 0인 경우에는 이전에 출력된 Q와 Q’의 값이 유지되며, RS 플립플롭은 R과 S가 동시에 1이 되는 것을 허용하지 않기 때문에 위와 같은 표가 완성된다.**

**RS 플립플롭은 NAND 게이트를 사용하여 설계할 수도 있고 NOR 게이트를 사용하여 설계할 수도 있다. 먼저 NAND 게이트를 사용하여 설계한 RS 플립플롭을 살펴보고자 한다. RS 플립플롭은 앞서 얘기했던 바와 같이 입력 r, s, Clk와 출력 q, qn으로 구성되어 있다. 입력과 출력을 정의하기 위한 Verilog code는 다음과 같다.**

**module sr\_nand(**

**input s, r, Clk,**

**output q, qn**

**);**

**이제 q와 q not을 각각 정의해야 하는데, 두 출력은 각각 두개의 nand 게이트로 구성되어 있으며, 두 출력을 정의하기 위해서는 서로가 nand 게이트에 입력되어야 한다. 각각을 정의한 Verilog code는 다음과 같다.**

**assign qn = ~(~(Clk & r) & q);**

**assign q = ~(~(Clk & s) & qn);**

**q not은 Clk와 r이 모두 1일 때 1이 출력된다. Clk와 r을 입력 받는 NAND 게이트의 값은 두 입력이 모두 1일 때 0이 되고, NAND 게이트에서는 두 입력 중 하나만 0이어도 1이 출력되기 때문에 q not의 값이 1이 된다. 반면 q는 Clk와 s가 모두 1일 때 1이 출력된다. 마찬가지로, Clk와 s를 입력 받는 NAND 게이트의 값은 두 입력이 모두 1일 때 0이 되고 두 번째 NAND 게이트는 1을 출력한다. 이렇게 설계된 디자인 코드를 바탕으로 시뮬레이션 코드를 작성하면 다음과 같다.**

**module sr\_nand\_tb();**

**reg s, r, Clk;**

**wire q, qn;**

**sr\_nand u\_inv(**

**.s(s),**

**.r(r),**

**.Clk(Clk),**

**.q(q),**

**.qn(qn));**

**initial begin**

**s = 1'b0;**

**r = 1'b0;**

**Clk = 1'b1;**

**end**

**always @(s or r or Clk) begin**

**s <= #20 ~s;**

**r <= #30 ~r;**

**Clk <= #10 ~Clk;**

**end**

**endmodule**

**시뮬레이션 코드에서 사용될 입출력들을 디자인 코드에서 사용한 입출력들과 각각 연결한 뒤, 세 입력 s, r, Clk를 0, 0, 1로 초기화하고 일정 간격에 따라 변화하도록 코드를 작성하였다. 이 코드로 시뮬레이션 실행하면 다음과 같은 시뮬레이션 결과를 볼 수 있다.**

텍스트이(가) 표시된 사진

자동 생성된 설명

**시뮬레이션 결과의 예시는 s가 0이고, r이 1이고 clk가 1인 상황을 보여주고 있다. Clk가 1이기 때문에 플립플롭이 작동되어야 하고, r의 값이 1이고, s의 값이 0이기 때문에 q가 0, qn의 값이 1이 되어야 하는데, 시뮬레이션 결과를 보면 두 출력 값이 정상적으로 출력되고 있음을 볼 수 있다.**

**다음으로, NOR 게이트를 사용해 RS 플립플롭을 설계한 과정을 살펴보고자 한다. NOR 게이트를 사용한 플립플롭은 두 개의 AND 게이트와 2개의 NOR 게이트로 설계되어 있으며 NAND 게이트를 사용한 RS 플립플롭과 동일하게 입력 S, R, Clk와 출력 q, qn로 구성되어 있다. 입력과 출력을 선언한 코드는 다음과 같다.**

**module sr\_nor(**

**input s, r, Clk,**

**output q, qn**

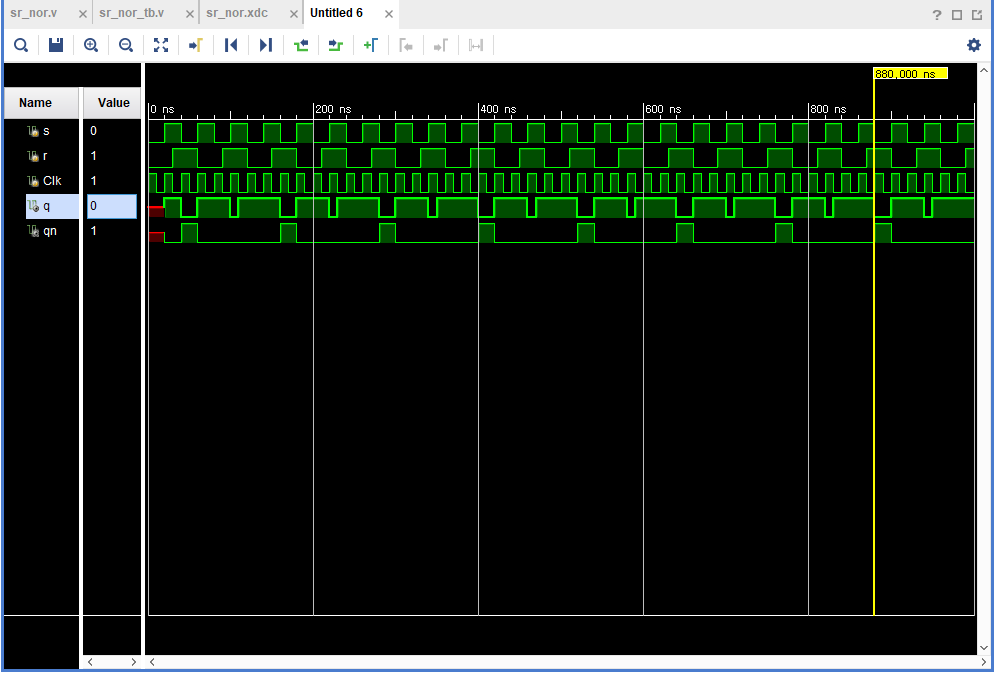
**);**

**이제 입력과 출력을 사용하여 q와 qn을 정의해야 한다. 여기서도 마찬가지로 qn을 정의할 때, q의 값이 입력되어야 하고, q를 정의할 때는 qn의 값이 입력되어야 한다. 두 출력에 대한 코드는 다음과 같다.**

**assign q = ~((r & Clk) | qn);**

**assign qn = ~((s & Clk) | q);**

**위 코드가 q와 qn을 정의한 코드이다. S의 값이 1이면, R은 0이 된다. 그러면 S & clk의 출력은 1이 되고, R & clk의 출력은 0이 된다. S & clk의 값이 1이면 ~((S & Clk) | q)의 값은 항상 0이 되기 때문에 qn은 0이 출력된다. Qn의 값이 0이면 q를 출력하는 NOR 게이트의 입력이 모두 0이 되기 때문에 q는 1이 출력된다. 반대로, S의 값이 0이면 R은 1이 된다. 그로 인해 S & clk의 출력은 0이 되고, R & clk의 값은 1이 된다. R & clk의 값이 1이 되면 q의 값은 항상 0이 되고, qn을 출력하는 NOR 게이트의 입력이 모두 0이 되기 때문에 qn은 1이 된다. 이제 이 디자인 코드를 바탕으로 시뮬레이션 코드를 작성하면 다음과 같다.**



**시뮬레이션 결과에서 보여지는 예시는 R이 1이고, S가 0이고, CLK가 1인 경우이다. CLK가 1이기 때문에 RS 플립플롭이 작동할 것이고, R의 값이 1이고, S의 값이 0이기 때문에 디자인 코드에 작성한 코드에 따라 qn의 값이 1이 되고, q의 값은 0이 되어야 한다. 시뮬레이션 결과 두 출력 값이 올바르게 출력되었기 때문에 NOR 게이트를 사용한 RS 플립플롭이 정상적으로 동작한다는 것을 확인할 수 있다.**

**2.**

**D 플립플롭은 RS 플립플롭을 바탕으로 설계된 플립플롭이다. D 플립플롭은 CLK와 D가 입력되고 Q와 Q’이 출력된다. 이 때 입력 D로 출력이 결정되는데, D가 1이면 S가 1이고, R이 0인 RS 플립플롭의 결과와 동일하게 1이 출력되고, D가 0이면 S가 0이고, R이 1인 RS 플립플롭의 결과와 동일하게 0이 출력된다. 또한 CLK가 1인 경우에만 플립플롭이 동작하고, CLK가 0인 경우에는 플립플롭이 동작하지 않는다. D 플립플롭의 behavioral table은 다음과 같다.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Clk** | **D** | **Q** | **Q’** |
| **0** | **X** | **Q** | **Q’** |
| **1** | **0** | **0** | **1** |
| **1** | **1** | **1** | **0** |

**CLK가 0일 때는 D의 값과 상관없이 플립플롭이 동작하지 않기 때문에 출력 값이 변화하지 않고, CLK가 1일 때 D가 0이면 Q가 0, Q’이 1이되며 D가 1일 때는 Q가 1, Q’이 0이 되어 위와 같은 표가 완성된다. D 플립플롭은 하나의 입력 d와 clock pulse가 입력되며, 두 출력 q와 qn이 출력된다. 이를 Verilog code로 작성하면 다음과 같다.**

**module dff(**

**input d, Clk,**

**output q, qn**

**);**

**이제 q와 qn을 정의해야 하는데 NOR 게이트를 사용하여 설계한 RS 플립플롭과 동일하게 두개의 AND 게이트와 두개의 NOR 게이트로 이루어져 있다.**

**assign q = ~((~d & Clk) | qn);**

**assign qn = ~((d & Clk) | q);**

**위 코드가 q와 qn을 정의한 코드이다. D의 값이 1이면, d’은 0이 된다. 그러면 d & clk의 출력은 1이 되고, ~d & clk의 출력은 0이 된다. d & clk의 값이 1이면 ~((d & Clk) | q)의 값은 항상 0이 되기 때문에 qn은 0이 출력된다. Qn의 값이 0이면 q를 출력하는 NOR 게이트의 입력이 모두 0이 되기 때문에 q는 1이 출력된다. 반대로, d의 값이 0이면 d’은 1이 된다. 그로 인해 d & clk의 출력은 0이 되고, ~d & clk의 값은 1이 된다. ~d & clk의 값이 1이 되면 q의 값은 항상 0이 되고, qn을 출력하는 NOR 게이트의 입력이 모두 0이 되기 때문에 qn은 1이 된다. 이제 이 디자인 코드를 바탕으로 시뮬레이션 코드를 작성하면 다음과 같다.**

**module dff\_tb();**

**reg d, Clk;**

**wire q, qn;**

**dff u\_inv(**

**.d(d),**

**.Clk(Clk),**

**.q(q),**

**.qn(qn));**

**initial begin**

**d = 1'b1;**

**Clk = 1'b1;**

**end**

**always @(d or Clk) begin**

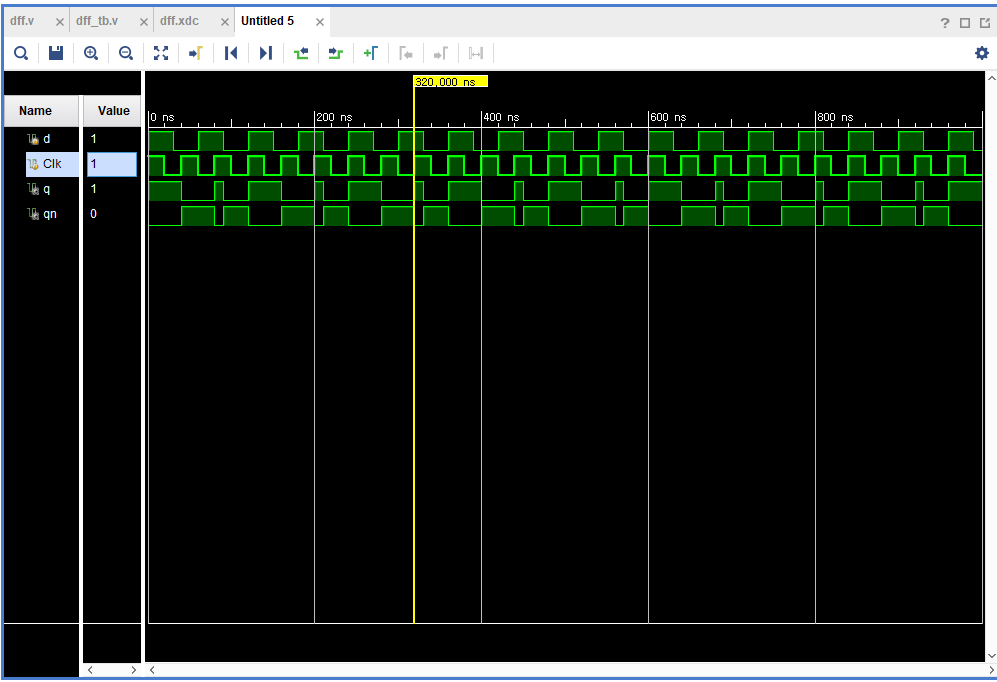
**d <= #30 ~d;**

**Clk <= #20 ~Clk;**

**end**

**endmodule**

**디자인 코드의 입력 D, Clk와 출력 q, qn을 각각 연결하고, d를 1로, clk를 1로 초기화한 뒤, 일정 간격에 따라 두 입력이 변화하도록 설계한다. 이 시뮬레이션 코드로 시뮬레이션을 작동시키면 그 결과는 다음과 같다.**



**이 시뮬레이션에서 보여지는 출력 예시는 d가 1이고, clk가 1인 경우이다. Clk가 1이기 때문에 D 플립플롭이 작동할 것이고, d가 1이기 때문에 RS 플립플롭에서 S가 1이고 R이 0일 때의 결과와 동일하게 q의 값이 1이 출력되고, qn의 값이 0이 출력될 것이다. 위에서 보여지는 시뮬레이션 결과가 이와 동일한 값을 출력하고 있기 때문에 이번 실험에서 설계한 D 플립플롭이 정상적으로 작동한다는 것을 확인할 수 있다.**

**3.**

**이번 실험에서는 NAND 게이트를 사용한 RS 플립플롭과 NOR 게이트를 사용한 RS 플립플롭, 그리고 D 플립플롭을 설계하였다. D 플립플롭이 RS 플립플롭을 응용하여 설계된 만큼 D 플립플롭을 설계하는 코드와 RS 플립플롭을 설계하는 방식이 유사하다는 것을 확인할 수 있었다. RS 플립플롭은 NAND 게이트와 NOR 게이트를 사용한 두 가지 방법으로 설계할 수 있었는데, 출력 결과는 두 가지 방법이 동일하였으므로 모두 올바르게 설계된 것을 확인할 수 있었다. RS 플립플롭은 (0,0), (1,0), (0,1) 세 가지 입력이 가능하였는데, 0, 0은 이전 출력이 그대로 유지되는 결과를 출력했으며, S가 1인 경우에는 Q가 1, R이 1인 경우에는 Q가 0이라는 결과를 출력하였다. D 플립플롭은 D가 0 또는 1인 두 가지 입력이 가능하였는데, D가 1인 것은 S가 1인 것과 동일하게 1이 출력되는 결과를 얻었고, D가 0인 것은 R이 1인 것과 동일하게 0이 출력되는 결과를 얻었다. 이번 실험에서 설계한 RS 플립플롭과 D 플립플롭 외에도 JK 플립플롭과 T 플립플롭이 존재하는데, JK 플립플롭 역시 RS 플립플롭을 응용하여 설계할 수 있고, RS 플립플롭과의 차이점은 1, 1이라는 입력이 가능하다는 것이다. T 플립플롭은 1이 입력되면 출력 값이 반전되고, 0이 입력되면 출력 값이 동일하게 유지되는 플립플롭이다. 이번 실험에서 설계한 두 플립플롭 외에 JK 플립플롭과 T 플립플롭의 설계에 대해서도 고민해볼 필요가 있다.**

**4.**

**플립플롭에는 여기표라는 게 존재한다. 영어로는 excitation table이라고 하며, 이는 이전 출력과 현재 출력을 통해 역으로 입력의 값을 도출하는 표이다. RS 플립플롭을 예로 들어보자. 출력 Q가 0에서 0으로 유지되었다면 가능한 S와 R의 값은 0, 0 혹은 0, 1이다. R이 1이 되어 Q가 0이 출력되었을 수도 있고 두 입력이 모두 0이어서 출력 값이 변하지 않았을 수도 있기 때문이다. 따라서 이 때 S, R을 (0, X)라고 표시한다. Q가 0에서 1로 변화하였을 때는 S가 1이 되는 하나의 경우 밖에 없다. 따라서 이 때 S, R을 (1, 0)으로 표시한다. Q가 1에서 0으로 변화하였을 때 역시 R이 1이 되는 하나의 경우 밖에 없으므로 S, R이 (0, 1)이라는 것을 도출할 수 있다. Q가 1에서 1로 유지되었을 때는 S, R이 1, 0 혹은 0,0이 될 수 있다. S가 1이 되어 Q가 1이 출력될 수도 있고, 두 입력이 모두 0이 되어 출력 값이 유지될 수도 있기 때문이다. 따라서 이 때 S, R을 (X, 0)이라고 표시한다. 이와 같이 출력 Q의 변화를 통해 S와 R의 값을 역으로 도출해 내는 표가 RS 플립플롭의 excitation table이라고 하며, RS 플립플롭뿐 아니라 다른 플립플롭도 동일한 방식으로 excitation table을 도출할 수 있다.**