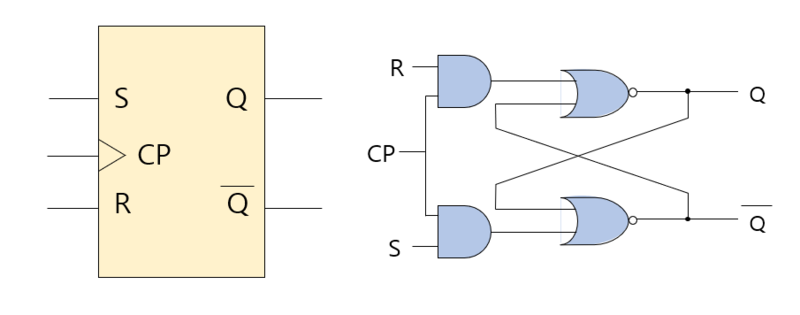
11주차 결과보고서

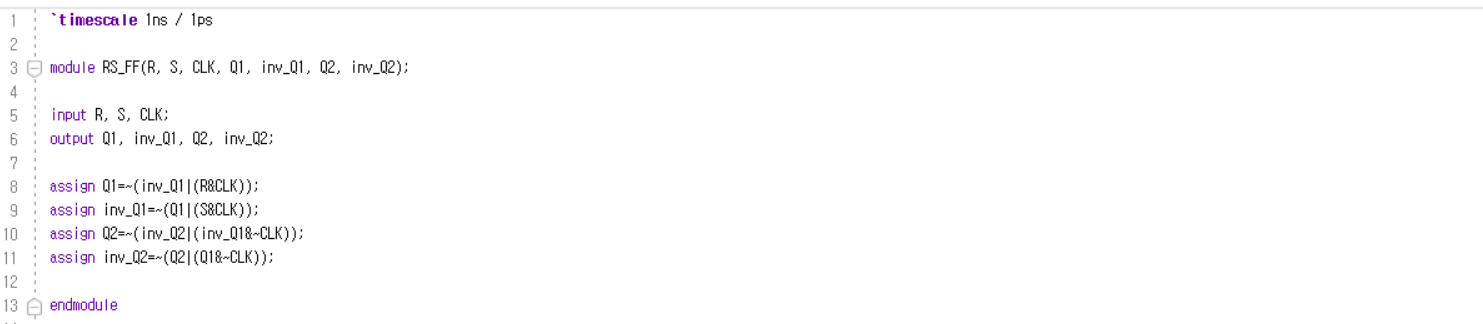
전공: 물리학과 학년: 3학년 학번: 20191286 이름: 김나현

1. RS 플립플롭의 결과 및 simulation 과정에 대해서 설명하시오(Verilog source, simulation 출력 예시, table 등의 과정을 상세히 적을 것).

RS 플립플롭은 다음 <Figure 1>과 같이 AND 게이트 두 개와 NOR 게이트 두 개로 구현이 가능하며 Q는 R과 CLK을 AND 연산한 것과 Q’의 NOR 연산으로 구할 수 있고 Q’은 S와 CLK을 AND 연산한 것과 Q의 NOR 연산으로 구할 수 있습니다. <Figure 2>의 Verilog 코드는 이러한 RS 플립플롭을 이용하여 Master-Slave RS 플립플롭을 구현한 것으로, Q1과 inv\_Q1은 각각 Master RS 플립플롭에서의 출력 Q와 Q’을 의미하고 Q2와 inv\_Q2는 각각 Slave RS 플립플롭에서의 출력 Q와 Q’을 의미합니다.

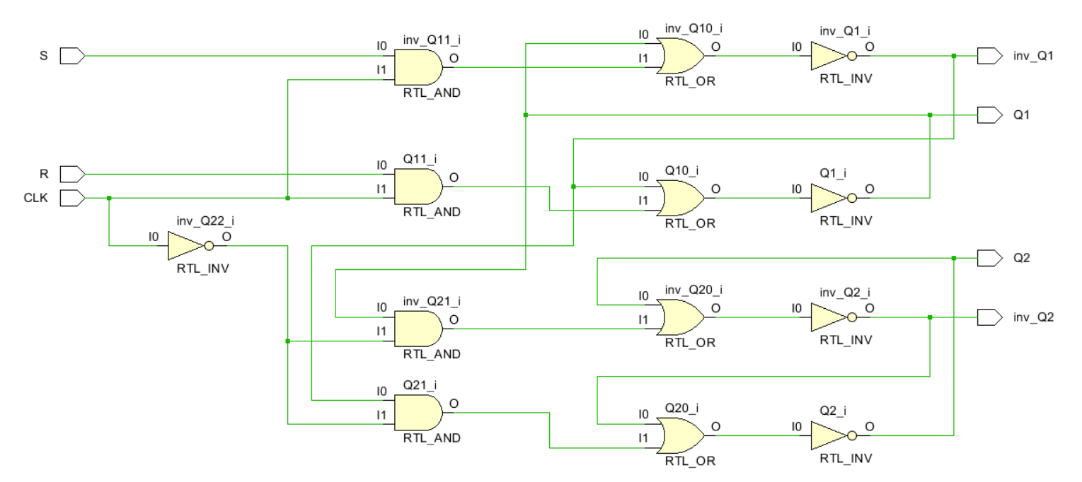


<Figure 1> RS 플립플롭

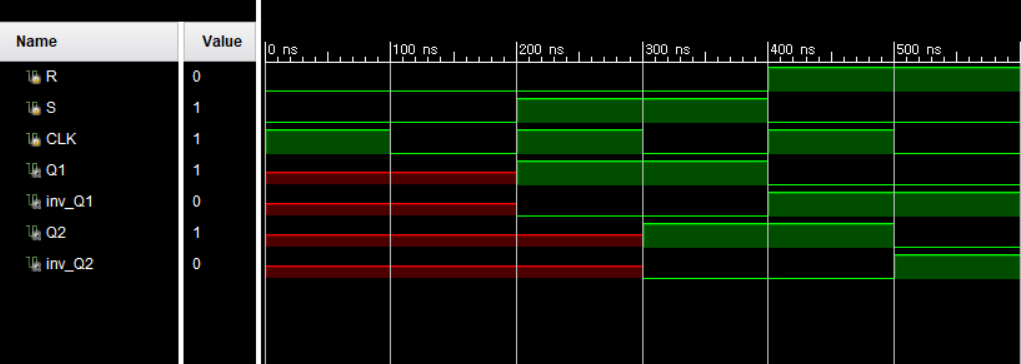


<Figure 2> RS 플립플롭 Verilog 코드

따라서, <Figure 2>의 Verilog 코드의 schematic을 확인해보면 다음 <Figure 3>과 같고 <Figure 2>에서 NOR 연산 대신 NOT과 OR 연산을 이용하여 코드를 구현하였으므로 RS 플립플롭이 AND 게이트와 NOT 게이트, OR 게이트로 구현된 것을 확인할 수 있습니다. 또한, Master-Slave 플립플롭에서는 두 개의 플립플롭이 각각 다른 edge에서 trigger되어야 하므로 Master RS 플립플롭은 상승 에지에서 trigger되며 Slave RS 플립플롭은 하강 에지에서 trigger되도록 코드를 작성하였습니다.



<Figure 3> RS 플립플롭 schematic diagram

test bench 코드를 작성하여 <Figure 2>의 simulation 결과를 확인해보면 다음 <Figure 4>와 같습니다.

<Figure 4> RS 플립플롭 simulation 결과

위의 simulation 결과를 보면 R과 S가 모두 0일 때 상승 에지에서는 Q1과 Q1’이 이전의 값을 그대로 출력해야 하지만 이전의 값이 없었으므로 unknown을 뜻하는 X를 출력하였고, 하강 에지에서는 Q1과 Q1’이 X이므로 Q2와 Q2’도 X를 출력하는 것을 확인할 수 있습니다. R이 0, S가 1이 되면 상승 에지에서는 Q1이 1로 Set되고, Q1’은 0이 되며 Slave RS 플립플롭은 trigger되지 않기 때문에 이전의 값, X를 그대로 출력하며 하강 에지에서는 Slave RS 플립플롭만이 trigger되어 Q1과 Q1’이 그대로 전달되어 Q2와 Q2’에서 출력되는 것을 확인할 수 있습니다. R이 1, S가 0이 되면 상승 에지에서는 Q1이 0으로 Reset되고, Q1’은 1이 되며 Q2와 Q2’는 변함없이 이전의 값을 출력하는 것을 확인할 수 있고 하강 에지에서는 Q1과 Q1’이 그대로 전달되어 Q2가 0, Q2’이 1을 출력하는 것을 확인할 수 있습니다. RS 플립플롭은 R와 S이 모두 1이 되는 경우는 사용하지 않으므로 본 simulation 결과에서도 역시 R과 S가 모두 1인 경우는 다루지 않았습니다.

다음 <Table 1>은 <Figure 4>의 simulation 결과를 토대로 작성한 RS 플립플롭의 table입니다. 이 표는 위에서부터 순서대로 나타난 것이며 CLK이 0인 것은 하강 에지를, CLK이 1인 것은 상승 에지를 의미합니다.

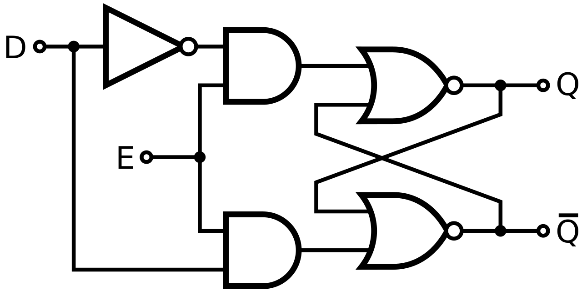
<Table 1> RS 플립플롭 table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| R | S | CLK | Q1 | Q1’ | Q2 | Q2’ |
| 0 | 0 | 1 | X | X | X | X |
| 0 | 0 | 0 | X | X | X | X |
| 0 | 1 | 1 | 1 | 0 | X | X |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 |

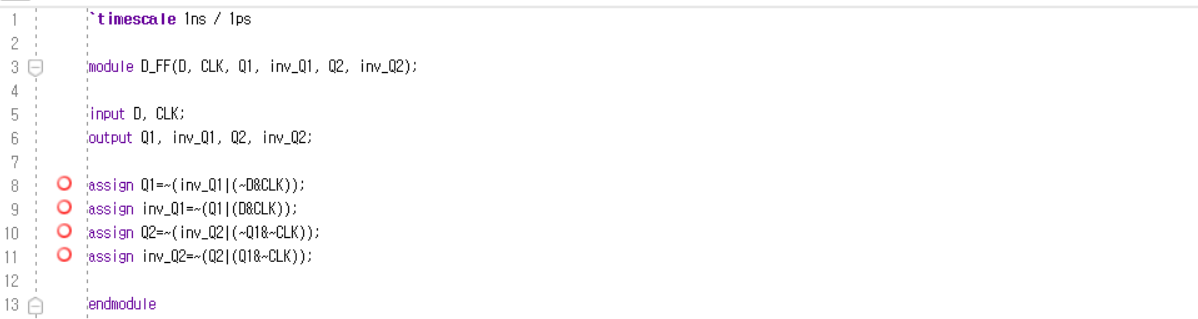
Clock 펄스가 반복되면서 상승 에지, 하강 에지가 나타날 때마다 각각 Master, Slave RS 플립플롭의 상태 변화를 촉발하는데 상승 에지로 인해 Master RS 플립플롭의 출력이 변화하면 하강 에지가 나타냈을 때는 Master RS 플립플롭의 출력이 그대로 Slave RS 플립의 출력에 반영되는 모습을 관찰할 수 있습니다.

1. D 플립플롭의 결과 및 simulation 과정에 대해서 설명하시오(Verilog source, simulation 출력 예시, table 등의 과정을 상세히 적을 것).

D 플립플롭은 다음 <Figure 5>와 같이 RS 플립플롭와 동일하게 AND 게이트 두 개와 NOR 게이트 두 개에 추가적으로 NOT 게이트를 하나 사용하여 구현이 가능하며 Q는 D’과 CLK을 AND 연산한 것과 Q’의 NOR 연산으로 구할 수 있고 Q’은 D와 CLK을 AND 연산한 것과 Q의 NOR 연산으로 구할 수 있습니다. <Figure 6>의 Verilog 코드는 Master-Slave D 플립플롭을 구현한 것으로, Q1과 inv\_Q1은 각각 Master D 플립플롭에서의 출력 Q와 Q’을 의미하고 Q2와 inv\_Q2는 각각 Slave D 플립플롭에서의 출력 Q와 Q’을 의미합니다.

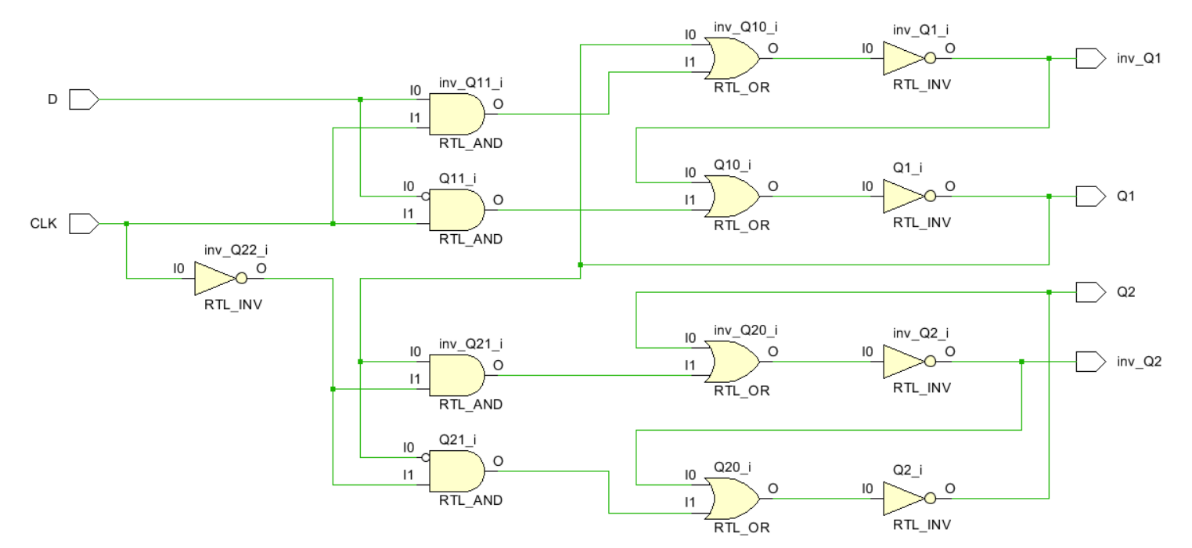


<Figure 5> D 플립플롭



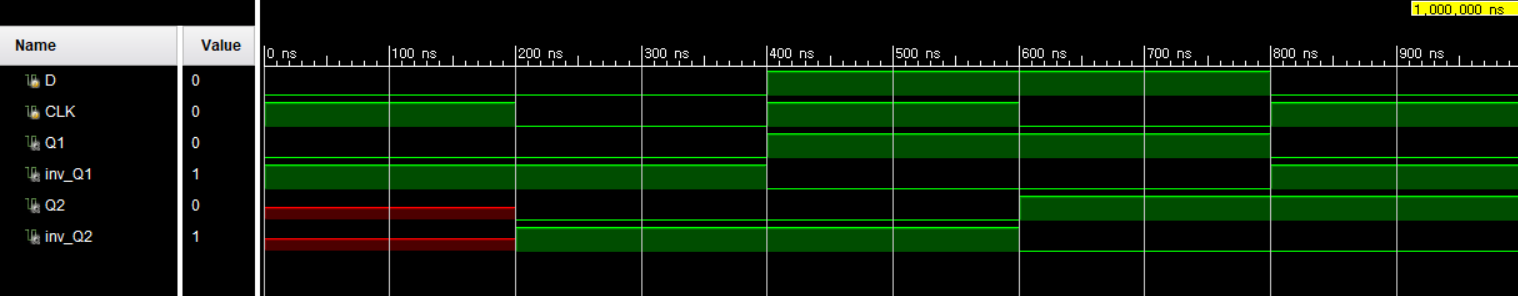
<Figure 6> D 플립플롭 Verilog 코드

따라서, <Figure 6>의 Verilog 코드의 schematic을 확인해보면 다음 <Figure 7>과 같고 <Figure 6>에서 NOR 연산 대신 NOT과 OR 연산을 이용하여 코드를 구현하였으므로 D 플립플롭이 AND 게이트와 NOT 게이트, OR 게이트로 구현된 것을 확인할 수 있습니다. 또한, Master-Slave 플립플롭에서는 두 개의 플립플롭이 각각 다른 edge에서 trigger되어야 하므로 Master D 플립플롭은 상승 에지에서 trigger되며 Slave D 플립플롭은 하강 에지에서 trigger되도록 코드를 작성하였습니다.



<Figure 7> D 플립플롭 schematic diagram

test bench 코드를 작성하여 <Figure 6>의 simulation 결과를 확인해보면 다음 <Figure 8>과 같습니다.



<Figure 8> D 플립플롭 simulation 결과

위의 simulation 결과를 보면 D가 0일 때 RS 플립플롭에서 S가 0이고 R이 1인 것과 동일하므로 상승 에지에서는 Q1이 0으로 Reset되고 Q1’이 1이 되며 Slave D 플립플롭은 아직 활성화되지 않았기 때문에 Q2와 Q2’은 알 수 없는 값, 즉 X를 출력합니다. D가 0일 때 하강 에지에서는 Slave D 플립플롭만이 활성화되므로 Q2과 Q2’이 Q1과 Q1’을 그대로 반영하여 출력하는 것을 확인할 수 있습니다. D가 1이 되면 RS 플립플롭에서 S가 1이고 R이 0인 것과 동일하므로 상승 에지에서는 Q1이 1로 Set되고, Q1’은 0이 되며 Slave D 플립플롭은 trigger되지 않기 때문에 이전의 값을 그대로 출력하며 하강 에지에서는 Slave D 플립플롭만이 trigger되어 Q1과 Q1’이 그대로 전달되어 Q2와 Q2’에서 출력되는 것을 확인할 수 있습니다. 이처럼 D 플립플롭은 RS 플립플롭에서 R이나 S 중 하나만이 1이 되는 상황만이 나타나고 플립플롭이 trigger될 때마다 이전의 상태와 무관하게 D의 값이 그대로 Q에 반영되어 나타나는 것을 확인할 수 있습니다.

다음 <Table 2>은 <Figure 8>의 simulation 결과를 토대로 작성한 D 플립플롭의 table입니다. 이 표는 위에서부터 순서대로 나타난 것이며 CLK이 0인 것은 하강 에지를, CLK이 1인 것은 상승 에지를 의미합니다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| D | CLK | Q1 | Q1’ | Q2 | Q2’ |
| 0 | 1 | 0 | 1 | X | X |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |

<Table 2> D 플립플롭 table

Clock 펄스가 반복되면서 상승 에지, 하강 에지가 나타날 때마다 각각 Master, Slave D 플립플롭의 상태 변화를 촉발하는데 상승 에지로 인해 Master D 플립플롭의 출력이 변화하면 하강 에지가 나타냈을 때는 Master D 플립플롭의 출력이 그대로 Slave D 플립의 출력에 반영되는 모습을 관찰할 수 있습니다. 또한 D 플립플롭은 trigger를 통해 상태 변화가 촉발되면 이전의 값과 무관하게 입력 D에 의해 출력 Q가 결정되는 것을 확인할 수 있습니다.

1. 결과 검토 및 논의 사항

Simulation 결과를 통해, RS 플립플롭과 D 플립플롭 모두 위와 같이 Verilog 코딩을 하면 올바르게 동작한다는 것을 확인할 수 있었습니다. 두 플립플롭 모두 Master-Slave 플립플롭으로 구현하여서 두 개의 플립플롭이 각각 상승 에지, 하강 에지에서만 활성화되는 것을 관찰할 수 있었고 Slave 플립플롭이 활성화되면 Master 플립플롭의 결과가 그대로 반영되어 Slave 플립플롭의 출력으로 나타나는 것 또한 관찰이 가능했습니다. RS 플립플롭의 경우, R과 S가 모두 0으로 입력될 때 각 플립플롭에서의 출력이 이전과 동일한 것을 관찰 가능하지만 simulation에서 그러한 내용을 다루지 못했기 때문에 FPGA를 통해 R과 S에 모두 0이 입력되어 플립플롭이 이전 상태를 기억하고 그대로 반환하는 것을 관찰할 수 있지만 실습 시간 동안 Verilog 코딩과 FPGA 연결을 성공하지 못하여 사진을 첨부할 수 없었습니다. 또한 본 실습의 simulation을 통해서는 CLK 신호와 입력 데이터들이 동시에 변하여 두 플립플롭이 edge trigger에 의해 상태 변화가 촉발되는 것을 잘 관찰할 수 없었는데 Verilog 코드와 test bench를 수정하여 edge trigger에 의한 플립플롭 출력의 변화를 더욱 잘 관찰할 수 있게 할 수 있습니다.

1. 추가 이론 조사 및 작성

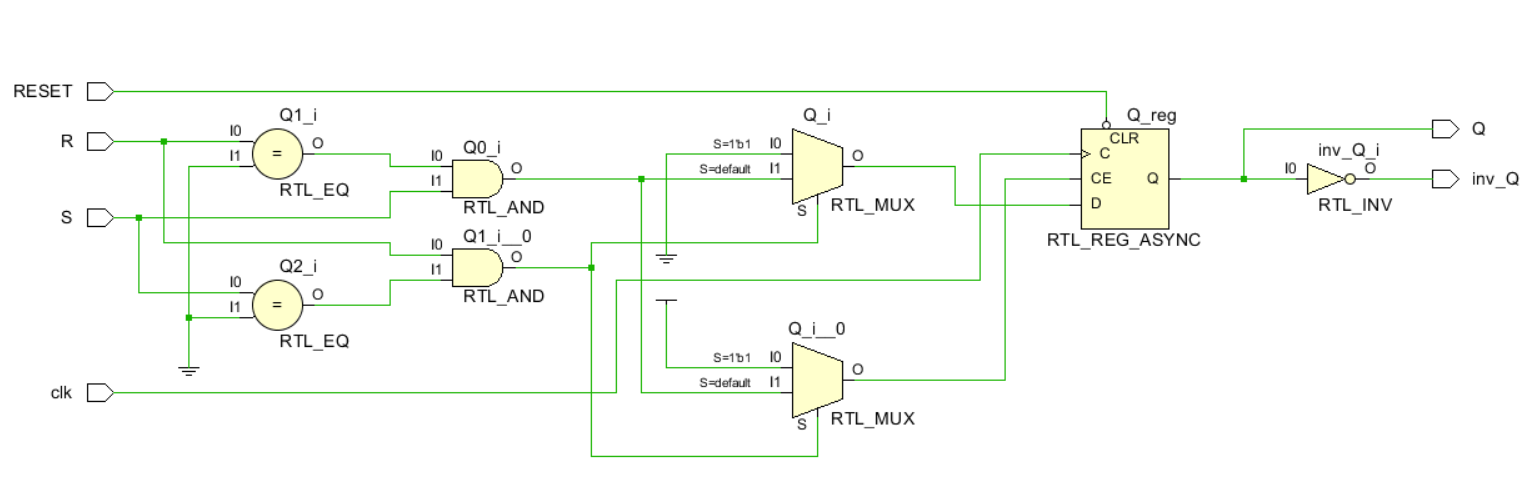
앞서 설명한 플립플롭의 코드들은 논리 게이트, 논리 연산을 이용하여 구현하였지만 추가 이론 조사에서는 always문을 이용하여 RS 플립플롭과 D 플립플롭을 구현하는 방법에 대해 다뤄보겠습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<Figure 9> always문을 이용한 RS 플립플롭 구현

위의 <Figure 9>는 always문을 이용하여 RS 플립플롭을 구현한 verilog 코드입니다. 이 코드를 이용하여 FPGA에 연결하면 RS 플립플롭이 상승 에지에서 trigger되는 것을 관찰할 수 있습니다.



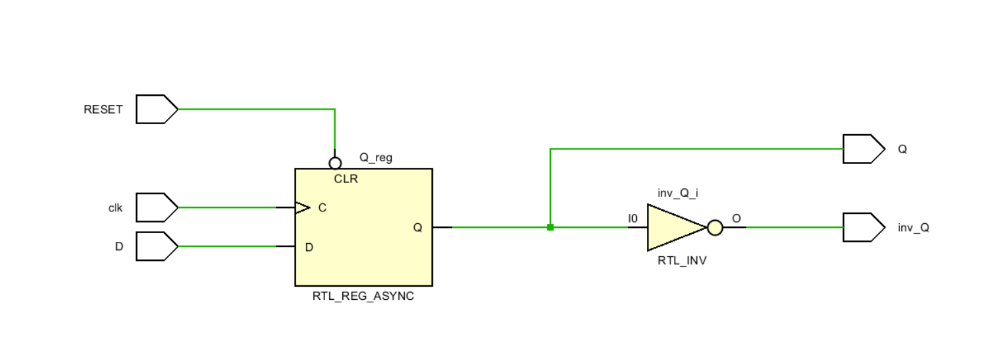
<Figure 10> always문을 이용한 RS 플립플롭 schematic diagram

<Figure 10>은 <Figure 9>의 schematic을 확인한 결과입니다.



<Figure 11> always문을 이용한 D 플립플롭 구현

위의 <Figure 9>는 always문을 이용하여 D 플립플롭을 구현한 verilog 코드입니다. 마찬가지로, 이 코드를 이용하여 FPGA에 연결하면 D 플립플롭이 상승 에지에서 trigger되는 것을 관찰할 수 있습니다.



<Figure 12> always문을 이용한 D 플립플롭 schematic diagram

<Figure 12>은 <Figure 11>의 schematic을 확인한 결과입니다.