**11주차 결과보고서**

**전공: 생명과학과 학년: 4학년 학번: 20182186 이름: 김승원**

**1. RS Flip-Flop의 결과와 simulation 과정 설명(Verilog source, simulation 출력 예시, table 등 과정 상세히)**

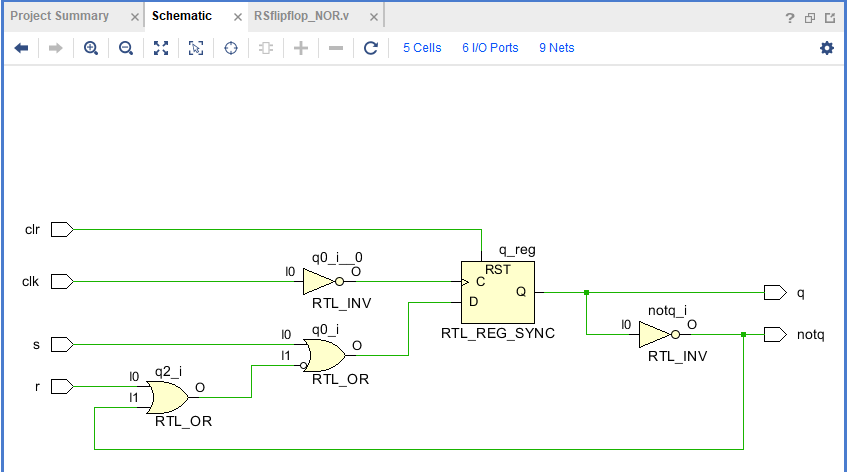
텍스트이(가) 표시된 사진

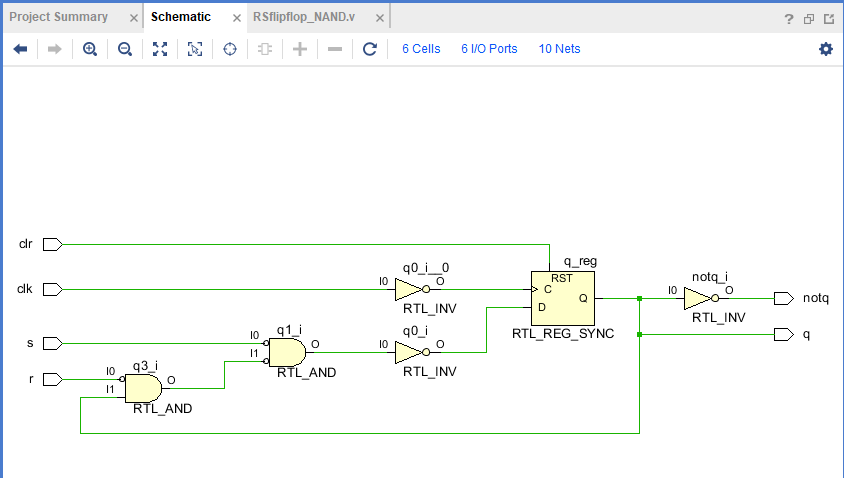
자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

**<Figure 1. NOR(좌), NAND(우)로 구성한 RS flipflop의 Verilog source code>**

RS flipflop은 Fig 1과 같이 source code를 통해 구현했다. If 문은 always 구문안에서 사용해야 하며, @(posedge !clk)를 통해 falling edge에서 업데이트 될 수 있도록 구현했다. 즉, @(posedge !clk)가 될 때 begin 하단의 명령을 실행하도록 했다. If(clr)문을 통해서 reset을 하도록 했고, 하단의 q <= s | ~(r|~q)를 통해서 식을 구현했다. 그리고 q의 보수인 q’을 notq로 할당했다. NAND로 구현한 RS flipflop 또한 logic은 같으므로, else문 안의 식만 변경해주었다.





**<Figure 2. RS flip flop schematic NOR(상), NAND(하)>**

Fig 2은 위의 RS flip flop(NOR(상), NAND(하))의 schematic 결과이다. 확인한 결과 NOR gate와 NAND gate로 잘 구현된 것을 확인했다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

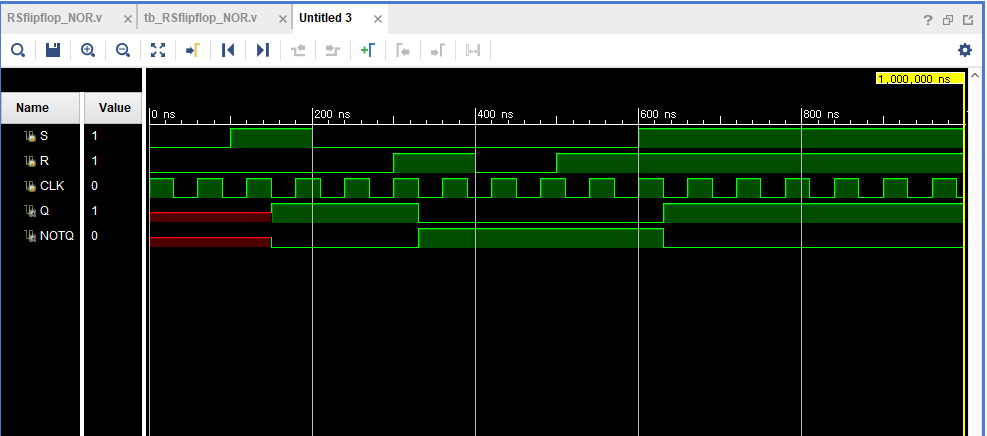
**<Figure 3. RS flipflop의 simulation을 위한 Test bench code>**

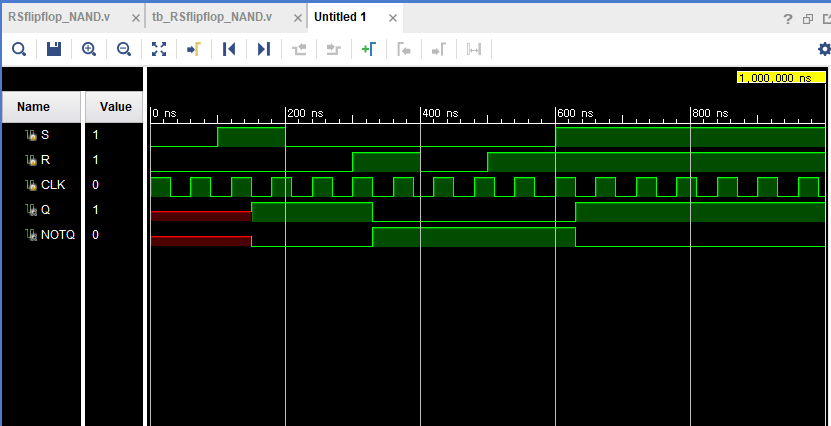
CLK 초기값을 1로 설정한 뒤 30ns마다 CLK 값에 변화를 주었다. 즉, 60ns를 한 주기로 했다. 그리고 S=1, R=0 부터 시작해서 S=1, R=1까지 순서대로 변화를 주었고, 위의 순서대로 simulation결과를 확인하고자 했다. 하단의 $monitor를 통해 TcI console창에 신호의 값이 변할 때 마다 출력하도록 했다.

**테이블이(가) 표시된 사진

자동 생성된 설명**

**<Figure 4. $monitor를 통해 확인한 결과>**





**<Figure 5. Simulation을 위한 RS flipflop(NOR, NAND) simulation>**

Fig 3는 Simulation을 위한 test bench code이고, Figure4는 각각 NAND gate로 구현한 RS flip flop과 NOR gate로 구현한 RS flip flop의 simulation 결과이다. Simulation 결과가 같은 이유는 logic은 동일하기 때문이다.

**<Table 1. RS flip flop의 진리표>**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | **Output** | |
| 순서 | R | S | Q | NOTQ |
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 1 |
| 6 | 1 | 1 | X | X |

Simulation 결과를 확인하여 진리표를 작성하였다. 진리표 순서대로 입력한 경우를 확인하였다. RS flip flop은 falling edge에서 q값이 업데이트 되도록 구현했기 때문에, falling edge시점에서 q를 확인하였다. Falling edge에서 q값이 업데이트가 잘 됐다.

결과적으로 순서1일 때, R=0, S=1인 경우 Q는 1로 입력이 됐다.

순서2에서는 R=0, S=0이므로, 이전 Q값이 유지될 것으로 예상했고, 결과적으로 Q=1이 나오며 이전 q값이 유지되는 것을 확인했다.

순서3의 경우 그리고 R=1, S=0인 경우에는 Q값이 0이 출력됐다.

순서4의 경우 R=0, S=0이므로, 이전 Q값이 유지됐다.

순서5의 경우 R=1, S=0으로, Q값이 0으로 출력됐다.

마지막으로 input R=0, S=0인 경우에는 이전 상태 Q값이 그대로 유지되는 것을 확인했다.

RS flip flop에서 R과 S가 모두 1인 경우는 don’t care에 해당하므로 어떤 값이든 나올수 있고, 무시하면 된다. 또한 NOTQ의 경우 Q의 보수이므로, 진리표에서 확인할 수 있듯이 값이 잘 출력됐다.

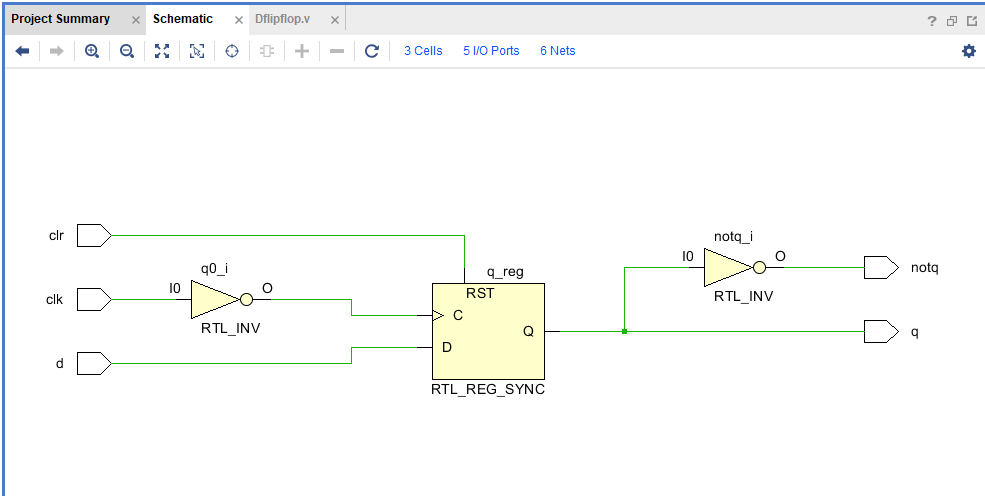
**2. D flip-flop의 결과 및 simulation 과정 설명(Verilog source, simulation 출력 예시, table등 과정 상세히 설명)**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**<Figure 6. D flipflop의 Verilog source code>**

D flipflop은 Fig 6 source code를 통해 구현했다. @(posedge !clk)를 통해 falling edge에서 업데이트 될 수 있도록 구현하였다. If(clr)문을 통해서 reset을 하도록 했고, 하단의 q <= d를 통해서 식을 구현했다. 그리고 q의 보수인 q’을 notq로 할당했다.



**<Figure 7. D flip flop의 schematic>**

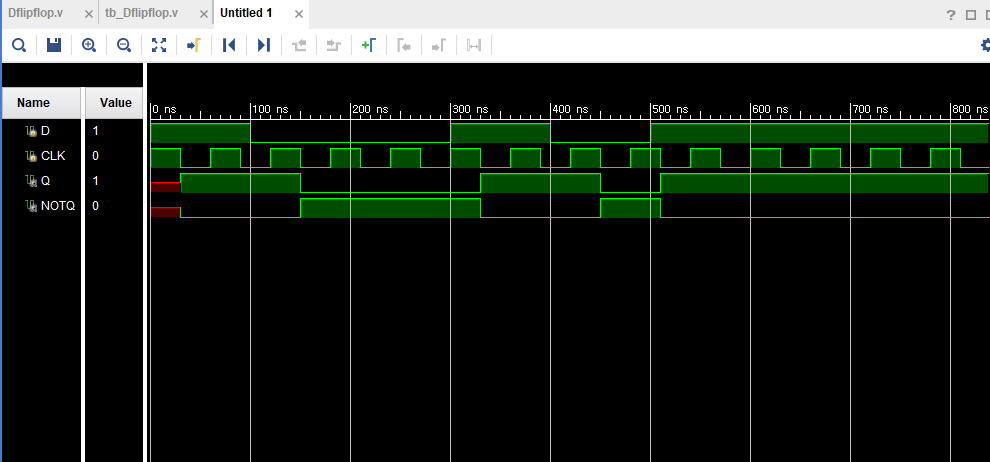
Fig 8은 위의 D flip flop의 schematic 결과이다. Source code가 정확함을 확인했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**<Figure 8. D flipflop simulation을 위한 test bench code>**

CLK는 30ns마다 0, 1로 바뀌도록 하여 주기를 60으로 하였고, D=0부터 D=1까지 총 6개의 D 입력값에 대한 Q 출력값을 확인하고자 위와 같이 code를 작성했다.

****

**<Figure 9. D flipflop의 simulation 결과>**

**<Table 2. D flip flop의 진리표>**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Input | Output | |
| 순서 | D | Q | Q’ |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 |
| 3 | 1 | 1 | 0 |
| 4 | 0 | 0 | 1 |
| 5 | 1 | 1 | 0 |
| 6 | 1 | 1 | 0 |

D flip flop의 falling edge에서 업데이트 되도록 구성했다. 따라서 falling edge에서 Q값의 변화를 관찰했다.

우선 초기 D값을 1로 설정했기 때문에 90ns falling edge시점에서 D값이 1이었기 때문에 다음 falling edge 시점까지 Q=1로 유지됐다.

순서1을 확인해보면, 다음 150ns의 falling edge시점에서 D=0이므로, Q=0이 된 것을 확인했다.

순서2는 D=0으로 D input값의 변화가 없었고, 이에 따라 Q값도 0으로 변화없이 유지됐다.

순서3은 D=1로 D input값이 변했다. 변한 이후 330ns까지 falling edge까지 변화가 없다가 falling edge에서 Q값이 1로 변했다.

순서4에서는 D=0이 됐고, Q값이 falling edge시점에 와서 Q=0이 되었다는 것을 확인했다.

순서5와 6에서는 D=1로 falling edge시점에서 Q=1로 값이 변했다. D값이 변화가 없었으므로, Q값 또한 1로 유지됐다.

**3. 결과 검토 및 논의 사항**

우선 RS flip flop에서는 S=1, R=0인 경우에는 Q output이 1이 출력됐다. 그리고 R=0, S=0으로 input의 변화가 있었을 때는 Q값이 유지될 것으로 예상했고, 확인한 결과 Q=1로 유지된 것을 확인했다. 또한 R=1, S=0으로 input의 변화를 주었을 때, Reset input이 들어가면서 Q값이 0으로 출력될 것으로 예상했고, 예상대로 0으로 출력됐다. 따라서 RS flipflop의 characteristic table과 excitation table은 다음과 같다.

**<Table 3. RS flip flop의 characteristic table>**

|  |  |  |
| --- | --- | --- |
| S | R | Q(next) |
| 0 | 0 | Q(이전 값 유지) |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | NA |

**<Table 4. RS flip flop의 excitation table>**

|  |  |  |  |
| --- | --- | --- | --- |
| Q | Q(next) | S | R |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | X | 0 |

**Characteristic equation: Q(next) = S + R’Q , SR=0**

D flip flop에서는 input D=0일 때, falling edge에서 값이 업데이트가 된 후 Q=0으로 출력됐다. D=1으로 입력된 경우 Q=1로 falling edge를 지난 후 출력됐다. 따라서 D flipflop의 characteristic table과 excitation table은 다음과 같다.

**<Table 5. D flip flop의 characteristic table>**

|  |  |
| --- | --- |
| D | Q(next) |
| 0 | 0 |
| 1 | 1 |

**<Table 6. D flip flop의 excitation table>**

|  |  |  |
| --- | --- | --- |
| Q | Q(next) | D |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Characteristic equation: Q(next) = D**

**4. 추가 이론 조사 및 작성**

**<T flip flop>**

**<Table 7. T flip flop의 characteristic table>**

|  |  |
| --- | --- |
| T | Q(next) |
| 0 | Q |
| 1 | Q’ |

**<Table 8. T flip flop의 excitation table>**

|  |  |  |
| --- | --- | --- |
| Q | Q(next) | T |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Characteristic equation: Q (next) = TQ’ + T’Q**

**<JK flip flop>**

**<Table 7. T flip flop의 characteristic table>**

|  |  |  |
| --- | --- | --- |
| J | K | Q(next) |
| 0 | 0 | Q |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | Q’ |

**<Table 8. T flip flop의 excitation table>**

|  |  |  |  |
| --- | --- | --- | --- |
| Q | Q(next) | J | K |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

**Characteristic equation: Q(next) = JQ’ + K’Q**