**11주차 결과 보고서**

남민혁

**1.** **RS Latch 의 결과 및 Simulation 과정에 대해서 설명하시오. (verliog source, 출력 예시, 과정 상세히 적을것!)**

RS Latch는 R(Reset)과 S(Set) 두개의 입력으로 두개의 출력 Q와 Q’ 값을 변화시키는 Latch의 한 종류이다. 구체적인 입력과 그에 따른 출력의 결과는 다음과 같다.

**-Truth Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| Enable | S | R |  |  |
| 0 | X | X |  |  |
| 1 | 0 | 0 |  |  |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | X | X |

Enable이 켜져있는 상태에서 S만 켜진 경우 다음 출력이 1, R만 켜진 경우 다음 출력이 0으로 설정된다. S와 R 모두 꺼져있는 경우에는 이전 출력이 그대로 출력되지만, 둘다 켜진 경우에는 정상적으로 실행되지 않는다.

**-Verilog 코딩**

RS Latch구현을 위하여 module rsllD(r, s, q, nq, en);로서 모듈을 선언하였으며, 각각의 입력에 따라 truth table에서 정해진 결과를 출력하도록 코딩하였다.

각 행은 논리회로 구성을 위한 Verilog 코드, 이를 바탕으로 모든 가능한 입력으로 부터 기대되는 출력이 나타나는지를 확인하기 위한 Test bench 코드이다. 끝으로 이를 FPGA에 이진 프로그램을 주입하여 입력을 받고, 출력 값을 LED 등에 나타내기 위한 Constraint 코드이다.

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module rsllD(r, s, q, nq, en);  input r, s, en;  output q, nq;  reg q, nq;  always@(posedge en) begin  case(s)  0: if (r == 0) begin  q <= q; nq <= nq;  end  else begin  q <= 0; nq <= 1;  end  1: if (r == 0) begin  q <= 1; nq <= 0;  end  endcase  end  endmodule |
| Test Bench | `timescale 1ns / 1ps  module rsllD\_tb();  reg r, s, en;  wire q, nq;  rsllD con(r, s, q, nq, en);  initial begin  en = 0; s = 1; r = 0;  s = #10 0;  r = #10 1;  r = #10 0;  s = #10 1; r = 1;  end  always en = #5 ~en;  endmodule |
| XDC | set\_property IOSTANDARD LVCMOS18 [get\_ports {r}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {s}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {en}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {q}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {nq}]  set\_property PACKAGE\_PIN J4 [get\_ports {r}]  set\_property PACKAGE\_PIN L3 [get\_ports {s}]  set\_property PACKAGE\_PIN K3 [get\_ports {en}]  set\_property PACKAGE\_PIN F15 [get\_ports {q}]  set\_property PACKAGE\_PIN F13 [get\_ports {nq}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {q}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {q\_OBUF}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {nq}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {nq\_OBUF}] |

**-Simulation 출력 결과 비교**

**Graphical user interface

Description automatically generated**

모든 가능한 입력 값에 대한 출력값을 확인하기 위하여 Simulation을 시행한 결과 의도한 진리표와 모든 결과가 동일함을 확인하였다.

**-출력 예시**

위의 시뮬레이션 결과는 5ns 주기로 enable 신호가 바뀔 때, R과 S 입력이 10ns 주기로 변경 될 때 q와 nq의 결과를 확인할 수 있다. 앞서 설명한 바와 같이 S만 1의 값을 가지는 경우에는 q가 1로 바뀌는 것을 확인할 수 있다. 이후, R과 S가 모두 0인 경우에는 q가 그대로 유지되었고, R만 1의 값을 가지는 경우에는 q가 0으로 바뀌는 것으로 나타났다. 끝으로, 모두 1의 값을 가지는 경우에는 이전의 결과가 그대로 유지되는 것으로 나타난다.

XDC 파일을 참고할 때, 각 스위치는 R, S, Enable를 의미하며, 그 결과인 q와 nq가 첫번째와 두번째 스위치 위에 있는 LED 등으로서 출력된다. R, S의 입력을 스위치로 입력한 후, Enable 스위치를 0에서 1로 바꾸어 주면, simulation에서 확인한 것과 동일한 결과를 확인할 수 있다. 예를 들어 nq의 LED에만 불이 들어와있는 경우 (q=0인 상황), R 스위치를 0으로 설정하고, S 스위치를 위로 올려 1로 설정한 후, Enable 스위치를 아래로 내렸다가 올리면 q LED에 불이 들어 오는 것을 확인할 수 있다.

**-구현된 Schematic**

Diagram, schematic

Description automatically generated

**2. RS Flip-Flop 의 결과 및 Simulation 과정에 대해서 설명하시오. (verliog source, 출력 예시, 과정 상세히 적을것!)**

RS Flip-Flop은 RS Latch와 동일하게 R(Reset)과 S(Set) 두개의 입력으로 두개의 출력 Q와 Q’ 값을 변화시키는 Flop-Flop의 한 종류이다. 유일한 차이는 기존 Enable bit이 Clock bit으로 구현되었다는 점이다. 구체적인 입력과 그에 따른 출력의 결과는 다음과 같다.,

**-Truth Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| CLK | S | R |  |  |
| 0 | X | X |  |  |
| 1 | 0 | 0 |  |  |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | X | X |

CLK이 켜져있는 상태에서 S만 켜진 경우 다음 출력이 1, R만 켜진 경우 다음 출력이 0으로 설정된다. S와 R 모두 꺼져있는 경우에는 이전 출력이 그대로 출력되지만, 둘다 켜진 경우에는 정상적으로 실행되지 않는다.

**-Verilog 코딩**

RS Flip-Flop 구현을 위하여 module rsffD(r, s, q, nq, clk);로서 모듈을 선언하였으며, 각각의 입력에 따라 truth table에서 정해진 결과를 출력하도록 코딩하였다.

각 행은 논리회로 구성을 위한 Verilog 코드, 이를 바탕으로 모든 가능한 입력으로 부터 기대되는 출력이 나타나는지를 확인하기 위한 Test bench 코드이다. 끝으로 이를 FPGA에 이진 프로그램을 주입하여 입력을 받고, 출력 값을 LED 등에 나타내기 위한 Constraint 코드이다.

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module rsffD(r, s, q, nq, clk);  input r, s, clk;  output q, nq;  reg q, nq;  always@(posedge clk) begin  case(s)  0: if (r == 0) begin  q <= q; nq <= nq;  end  else begin  q <= 0; nq <= 1;  end  1: if (r == 0) begin  q <= 1; nq <= 0;  end  endcase  end  endmodule |
| Test Bench | `timescale 1ns / 1ps  module rsfffD\_tb();  reg r, s, clk;  wire q, nq;  rsffD con(r, s, q, nq, clk);  initial begin  clk = 0; s = 1; r = 0;  s = #10 0;  r = #10 1;  r = #10 0;  s = #10 1; r = 1;  end  always clk = #5 ~clk;  endmodule |
| XDC | set\_property IOSTANDARD LVCMOS18 [get\_ports {r}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {s}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {clk}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {q}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {nq}]  set\_property PACKAGE\_PIN J4 [get\_ports {r}]  set\_property PACKAGE\_PIN L3 [get\_ports {s}]  set\_property PACKAGE\_PIN K3 [get\_ports {clk}]  set\_property PACKAGE\_PIN F15 [get\_ports {q}]  set\_property PACKAGE\_PIN F13 [get\_ports {nq}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {q}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {q\_OBUF}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {nq}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {nq\_OBUF}] |

**-Simulation 출력 결과 비교**

**A picture containing green, holding, game, sitting

Description automatically generated**

모든 가능한 입력 값에 대한 출력값을 확인하기 위하여 Simulation을 시행한 결과 의도한 진리표와 모든 결과가 동일함을 확인하였다.

**-출력 예시**

위의 시뮬레이션 결과는 5ns 주기로 CLK 신호가 바뀔 때, R과 S 입력이 10ns 주기로 변경 될 때 q와 nq의 결과를 확인할 수 있다. 앞서 설명한 바와 같이 S만 1의 값을 가지는 경우에는 q가 1로 바뀌는 것을 확인할 수 있다. 이후, R과 S가 모두 0인 경우에는 q가 그대로 유지되었고, R만 1의 값을 가지는 경우에는 q가 0으로 바뀌는 것으로 나타났다. 끝으로, 모두 1의 값을 가지는 경우에는 이전의 결과가 그대로 유지되는 것으로 나타난다.

XDC 파일을 참고할 때, 각 스위치는 R, S, CLK를 의미하며, 그 결과인 q와 nq가 첫번째와 두번째 스위치 위에 있는 LED 등으로서 출력된다. R, S의 입력을 스위치로 입력한 후, CLK 스위치를 0에서 1로 바꾸어 주면, simulation에서 확인한 것과 동일한 결과를 확인할 수 있다. 예를 들어 nq의 LED에만 불이 들어와있는 경우 (q=0인 상황), R 스위치를 0으로 설정하고, S 스위치를 위로 올려 1로 설정한 후, CLK 스위치를 아래로 내렸다가 올리면 q LED에 불이 들어 오는 것을 확인할 수 있다.

**-구현된 Schematic**

Diagram, schematic

Description automatically generated

**3. D Latch 의 결과 및 Simulation 과정에 대해서 설명하시오. (verliog source, 출력 예시, 과정 상세히 적을것!)**

D Latch는 D라는 하나의 입력으로 두개의 출력 Q와 Q’ 값을 변화시키는 Latch의 한 종류이다. 구체적인 입력과 그에 따른 출력의 결과는 다음과 같다.,

**-Truth Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | |
| Enable | D |  |  |
| 0 | 0 |  |  |
| 0 | 1 |  |  |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Enable이 켜져있는 상태에서 D의 값이 다음 출력으로 결정된다. 즉, D가 켜진 경우 다음 출력은 1, 아닌 경우는 0으로 설정된다. Enable이 꺼져 있는 경우에는 어떠한 입력도 반영되지 않는다.

**-Verilog 코딩**

D Latch 구현을 위하여 module dllD(d, q, nq, en);로서 모듈을 선언하였으며, 각각의 입력에 따라 truth table에서 정해진 결과를 출력하도록 코딩하였다.

각 행은 논리회로 구성을 위한 Verilog 코드, 이를 바탕으로 모든 가능한 입력으로 부터 기대되는 출력이 나타나는지를 확인하기 위한 Test bench 코드이다. 끝으로 이를 FPGA에 이진 프로그램을 주입하여 입력을 받고, 출력 값을 LED 등에 나타내기 위한 Constraint 코드이다.

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module dllD(d, q, nq, en);  input d, en;  output q, nq;  reg q, nq;  always@(posedge en) begin  if (d == 0) begin  q <= 0; nq <= 1; end  else begin  q <= 1; nq <= 0; end  end  endmodule |
| Test Bench | `timescale 1ns / 1ps  module dllD\_tb();  reg d, en;  wire q, nq;  dllD con(d, q, nq, en);  initial begin  en = 0; d = 0;  d = #10 1;  end  always en = #5 ~en;  endmodule |
| XDC | set\_property IOSTANDARD LVCMOS18 [get\_ports {d}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {en}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {q}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {nq}]  set\_property PACKAGE\_PIN J4 [get\_ports {d}]  set\_property PACKAGE\_PIN L3 [get\_ports {en}]  set\_property PACKAGE\_PIN F15 [get\_ports {q}]  set\_property PACKAGE\_PIN F13 [get\_ports {nq}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {q}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {q\_OBUF}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {nq}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {nq\_OBUF}] |

**-Simulation 출력 결과 비교**

**A picture containing green, sitting, player, holding

Description automatically generated**

모든 가능한 입력 값에 대한 출력값을 확인하기 위하여 Simulation을 시행한 결과 의도한 진리표와 모든 결과가 동일함을 확인하였다.

**-출력 예시**

위의 시뮬레이션 결과는 5ns 주기로 Enable 신호가 바뀔 때, D 입력이 10ns 주기로 변경 될 때 q와 nq의 결과를 확인할 수 있다. 앞서 설명한 바와 같이 D가 0인 경우에는 q가 0으로 바뀌는 것을 확인할 수 있다. 이후, D가 1의 값을 가지는 경우에는 q가 1으로 바뀌는 것으로 나타난다.

XDC 파일을 참고할 때, 각 스위치는 D, Enable를 의미하며, 그 결과인 q와 nq가 첫번째와 두번째 스위치 위에 있는 LED 등으로서 출력된다. D의 입력을 스위치로 입력한 후, Enable스위치를 0에서 1로 바꾸어 주면, simulation에서 확인한 것과 동일한 결과를 확인할 수 있다. 예를 들어 nq의 LED에만 불이 들어와있는 경우 (q=0인 상황), D 스위치를 위로 올려 1로 설정한 후, Enable스위치를 아래로 내렸다가 올리면 q LED에 불이 들어 오는 것을 확인할 수 있다.

**-구현된 Schematic**

Diagram, schematic

Description automatically generated

**4. D Flip-Flop 의 결과 및 Simulation 과정에 대해서 설명하시오. (verliog source, 출력 예시, 과정 상세히 적을것!)**

D Flip-Flop은 D-latch와 동일하게 D라는 하나의 입력으로 두개의 출력 Q와 Q’ 값을 변화시키는 Filp-Flop의 한 종류이다. 유일한 차이는 Enable bit이 Clock bit으로 구현되었다는 점이다. 구체적인 입력과 그에 따른 출력의 결과는 다음과 같다.,

**-Truth Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | |
| CLK | D |  |  |
| 0 | 0 |  |  |
| 0 | 1 |  |  |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

CLK가 켜져있는 상태에서 D의 값이 다음 출력으로 결정된다. 즉, D가 켜진 경우 다음 출력은 1, 아닌 경우는 0으로 설정된다. CLK가 꺼져 있는 경우에는 어떠한 입력도 반영되지 않는다.

**-Verilog 코딩**

D Flip-Flop 구현을 위하여 module dffD(d, q, nq, clk);로서 모듈을 선언하였으며, 각각의 입력에 따라 truth table에서 정해진 결과를 출력하도록 코딩하였다.

각 행은 논리회로 구성을 위한 Verilog 코드, 이를 바탕으로 모든 가능한 입력으로 부터 기대되는 출력이 나타나는지를 확인하기 위한 Test bench 코드이다. 끝으로 이를 FPGA에 이진 프로그램을 주입하여 입력을 받고, 출력 값을 LED 등에 나타내기 위한 Constraint 코드이다.

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module dffD(d, q, nq, clk);  input d, clk;  output q, nq;  reg q, nq;  always@(posedge clk) begin  if (d == 0) begin  q <= 0; nq <= 1; end  else begin  q <= 1; nq <= 0; end  end  endmodule |
| Test Bench | `timescale 1ns / 1ps  module dffD\_tb();  reg d, clk;  wire q, nq;  dffD con(d, q, nq, clk);  initial begin  clk = 0; d = 0;  d = #10 1;  end  always clk = #5 ~clk;  endmodule |
| XDC | set\_property IOSTANDARD LVCMOS18 [get\_ports {d}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {en}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {q}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {nq}]  set\_property PACKAGE\_PIN J4 [get\_ports {d}]  set\_property PACKAGE\_PIN L3 [get\_ports {en}]  set\_property PACKAGE\_PIN F15 [get\_ports {q}]  set\_property PACKAGE\_PIN F13 [get\_ports {nq}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {q}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {q\_OBUF}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {nq}]  set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets {nq\_OBUF}] |

**-Simulation 출력 결과 비교**

**A picture containing green, holding, player, sitting

Description automatically generated**

모든 가능한 입력 값에 대한 출력값을 확인하기 위하여 Simulation을 시행한 결과 의도한 진리표와 모든 결과가 동일함을 확인하였다.

**-출력 예시**

위의 시뮬레이션 결과는 5ns 주기로 CLK 신호가 바뀔 때, D 입력이 10ns 주기로 변경 될 때 q와 nq의 결과를 확인할 수 있다. 앞서 설명한 바와 같이 D가 0인 경우에는 q가 0으로 바뀌는 것을 확인할 수 있다. 이후, D가 1의 값을 가지는 경우에는 q가 1으로 바뀌는 것으로 나타난다.

XDC 파일을 참고할 때, 각 스위치는 D, CLK를 의미하며, 그 결과인 q와 nq가 첫번째와 두번째 스위치 위에 있는 LED 등으로서 출력된다. D의 입력을 스위치로 입력한 후, CLK스위치를 0에서 1로 바꾸어 주면, simulation에서 확인한 것과 동일한 결과를 확인할 수 있다. 예를 들어 nq의 LED에만 불이 들어와있는 경우 (q=0인 상황), D 스위치를 위로 올려 1로 설정한 후, CLK스위치를 아래로 내렸다가 올리면 q LED에 불이 들어 오는 것을 확인할 수 있다.

**-구현된 Schematic**

Diagram, schematic

Description automatically generated

**5. JK Flip-Flop 의 결과 및 Simulation 과정에 대해서 설명하시오. (verliog source, 출력 예시, 과정 상세히 적을것!)**

JK Flip-Flop은 (2)에서 살펴본 RS Filp-Flop과 유사한 Flip-Flop의 한 종류로서, K와 J라는 두가지 입력이 각각 R(Reset)과, S(Set)에 대응되어 두개의 출력 Q와 Q’ 값을 변화시킨다. 기존 RS Flip-Flop과 다르게 clock이 활성화 되어있을 때, J와 K가 모두 활성화 되어 있는 경우 기존 결과를 반전시키는 기능이 추가되었다는 점에서 차이가 있다. 구체적인 입력과 그에 따른 출력의 결과는 다음과 같다.,

**-Truth Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| CLK | J | K |  |  |
| 0 | X | X |  |  |
| 1 | 0 | 0 |  |  |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 |  |  |

CLK이 켜져있는 상태에서 J만 켜진 경우 다음 출력이 1, K만 켜진 경우 다음 출력이 0으로 설정된다. S와 R 모두 꺼져있는 경우에는 이전 출력이 그대로 출력된다. 끝으로 둘다 켜진 경우에는 이전 값이 반전된다.

**-Verilog 코딩**

JK Flip-Flop 구현을 위하여 module jkffD(q, qb, j, k, clk, reset);로서 모듈을 선언하였으며, 각각의 입력에 따라 truth table에서 정해진 결과를 출력하도록 코딩하였다.

각 행은 논리회로 구성을 위한 Verilog 코드, 이를 바탕으로 모든 가능한 입력으로 부터 기대되는 출력이 나타나는지를 확인하기 위한 Test bench 코드이다. 끝으로 이를 FPGA에 이진 프로그램을 주입하여 입력을 받고, 출력 값을 LED 등에 나타내기 위한 Constraint 코드이다.

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module jkffD(q, qb, j, k, clk, reset);  output q, qb;  input j, k, clk, reset;  reg q;  assign qb = ~q;  always @(posedge clk) begin  if (reset)  q <= 0;  else if (k==0 & j==0)  q <= q;  else if (k==0 & j==1)  q <= 1;  else if (k==1 & j==0)  q <= 0;  else  q <= ~q;  end  endmodule |
| Test Bench | `timescale 1ns / 1ps  module jkfffD\_tb();  reg j, k, clk, reset;  wire q, qb;  jkffD con(q, qb, j, k, clk, reset);  initial begin  clk = 0; reset = 1; j = 1; k = 0;  reset = #10 0;  j = #10 0;  k = #10 1;  k = #10 0;  j = #10 1; k = 1;  end  always clk = #5 ~clk;  endmodule |
| XDC | set\_property IOSTANDARD LVCMOS18 [get\_ports {j}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {k}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {clk}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {reset}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {q}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {qb}]  set\_property PACKAGE\_PIN J4 [get\_ports {j}]  set\_property PACKAGE\_PIN L3 [get\_ports {k}]  set\_property PACKAGE\_PIN K3 [get\_ports {clk}]  set\_property PACKAGE\_PIN M2 [get\_ports {reset}]  set\_property PACKAGE\_PIN F15 [get\_ports {q}]  set\_property PACKAGE\_PIN F13 [get\_ports {qb}]  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets {clk\_IBUF}] |

**-Simulation 출력 결과 비교**

**A picture containing green, holding, person, player

Description automatically generated**

모든 가능한 입력 값에 대한 출력값을 확인하기 위하여 Simulation을 시행한 결과 의도한 진리표와 모든 결과가 동일함을 확인하였다.

**-출력 예시**

위의 시뮬레이션 결과는 5ns 주기로 CLK 신호가 바뀔 때, J와 K입력이 10ns 주기로 변경 될 때 q와 nq의 결과를 확인할 수 있다. 앞서 설명한 바와 같이 J만 1의 값을 가지는 경우에는 q가 1로 바뀌는 것을 확인할 수 있다. 이후, J와 K가 모두 0인 경우에는 q가 그대로 유지되었고, K만 1의 값을 가지는 경우에는 q가 0으로 바뀌는 것으로 나타났다. 끝으로, 모두 1의 값을 가지는 경우에는 이전의 결과가 반전되는 것으로 나타난다.

XDC 파일을 참고할 때, 각 스위치는 J, K, CLK, Reset을 의미하며, 그 결과인 q와 nq가 첫번째와 두번째 스위치 위에 있는 LED 등으로서 출력된다. J, K, Reset의 입력을 스위치로 입력한 후, CLK 스위치를 0에서 1로 바꾸어 주면, simulation에서 확인한 것과 동일한 결과를 확인할 수 있다. 예를 들어 nq의 LED에만 불이 들어와있는 경우 (q=0인 상황), K 스위치를 0으로 설정하고, J 스위치를 위로 올려 1로 설정한 후, CLK 스위치를 아래로 내렸다가 올리면 q LED에 불이 들어 오는 것을 확인할 수 있다. 한편 이 상황에서 J와 K 스위치를 모두 0으로 설정하고 Reset 스위치를 위로 올린 후 CLK 스위치를 조작하면, 다시 초기 상태로 바뀌는 것을 확인할 수 있다. 끝으로, J와 K 모두 1로 스위치를 올리고, CLK 스위치를 조작하면, 기존 상태에서 결과가 반전된다.

**-구현된 Schematic**

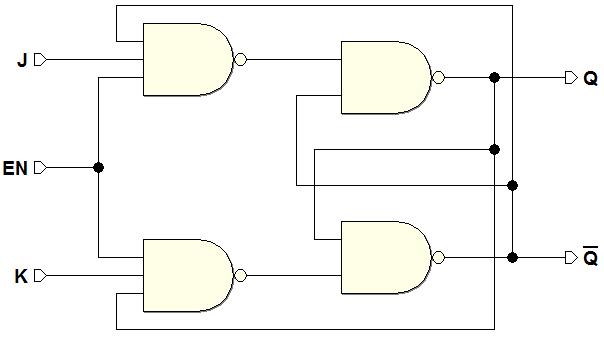
Diagram, schematic

Description automatically generated

**6. 결과 검토 및 논의 사항.**

이번주 실험에서는 D, RS, JK Flip-Flop과 D, RS Latch를 구현하였다. 진리표를 작성한 후 조건문을 사용하여 모든 가능한 상황에 대한 회로의 행동을 제어하는 방식으로 Verilog 코드를 작성하였다. 이후 모든 가능한 입력에 대해 simulate 하고, FPGA에 이진 프로그램을 주입하여 기대되는 결과가 나타나는지 확인하였다. 그 결과 모든 디자인이 의도한 진리표와 다르지 않고 동일하게 결과가 출력됨을 확인하였다. 또한 현재 FPGA에서는 clock이 자동적으로 시간을 세는 것이 아니라, 스위치를 통하여 clock의 신호를 입력받도록 함으로써, 사실상 Latch와 Flip-Flop 회로가 다르지 않았다.

**7. 추가 이론 조사 및 작성.**

본 실험에서 구현하지 않은 JK latch은 다음과 같은 특성을 가지고 있다. JK Latch는 JK flip-flop과 마찬가지로 SR latch에서 허용하지 않았던 입력 (S와 R모두 1로 활성화된 경우)를 허용하는 latch이다. 두 입력 모두 활성화된 경우에는 기존 출력 Q를 반전시킨다. 구체적으로 이를 구현하기 위한 논리회로는 다음과 같다. D latch에서와 마찬가지로, enable은 flip-flop에서 clock과 비슷한 역할을 하는 입력으로서, enable이 활성화 되어있을 때 들어온 입력에 대해서만 반응하고, 그 이외의 상황에 대해서는 해당 입력을 무시한다. JK latch의 진리표와 이를 NAND gate로 구현한 논리회로도는 아래와 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| Enable | J | K |  |  |
| 0 | X | X |  |  |
| 1 | 0 | 0 |  |  |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 |  |  |

**8. 참고 문헌**

강석태, “Verilog HDL Summary”, <http://vlsi.hongik.ac.kr/lecture/%EC%8B%A4%ED%97%98/Verilog_Summary.pdf>.

박재동, “[Verilog 예제] CLA (Carry Look Ahead) 모듈”, <http://egloos.zum.com/rabe/v/1285020>.

한국기술교육대학교 전기전자통신공학부, “디지털시스템 설계 및 실습” Verilog HDL 문법 강의자료, <https://cms3.koreatech.ac.kr/sites/yjjang/down/dsys11/M01_VerilogHDL01.pdf>.