**컴퓨터공학 설계 및 실험Ⅱ**

11주차 결과보고서

서강대학교 공학부 컴퓨터공학 전공

20171646 박태윤

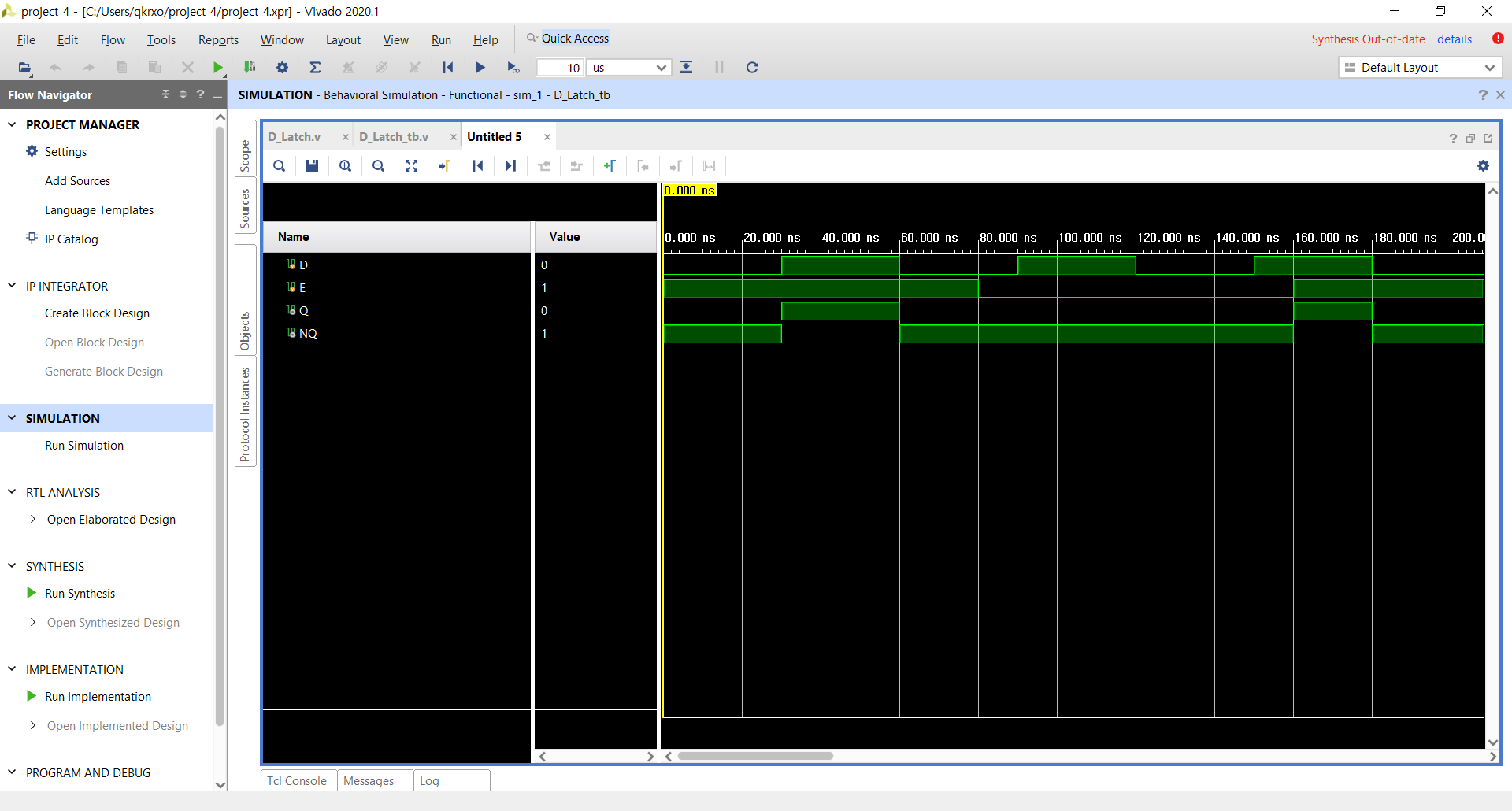
**1. D Latch의 결과 및 Simulation 과정에 대해서 설명하시오.**

D Latch를 다음과 같이 Verilog로 구현하였다.

|  |
| --- |
| **D\_Latch** |
| `timescale 1ns / 1ps  module D\_Latch(D, E, Q, NQ);  input D,E;  output Q, NQ;  reg Q, NQ;    initial begin  Q = 0;  NQ = 0;  end    always @(E or D) begin  if(E == 1)  begin  Q <= D;  NQ <= ~D;  end  end    endmodule |
| **D\_Latch\_tb** |
| `timescale 1ns / 1ps  module D\_Latch\_tb();  reg D,E;  wire Q, NQ;  D\_Latch connect(D, E, Q, NQ);  initial begin  D = 0; E = 1;  end  always D = #30 ~ D;  always E = #80 ~ E;  endmodule |

입력으로 D, E를 받으며 Q와 NQ를 출력한다. 실질적으로 입력 D에 따라 출력값이 바뀌며 E가 0일때는 D값에 상관없이 이전 출력을 유지하고 E가 1일 때에만 D에 따라 출력을 변화시킨다. E가 1일 때 Q는 D와 같은 값을 나타내며 NQ는 Q와 반대되는 값을 나타낸다.

구현한 Verilog코드를 이용하여 Simulation을 하였다.



출력 Q와 NQ가 항상 반대되는 값을 나타내며, E가 1일 때 D에 따라 Q와 NQ가 바뀌는 것을 확인할 수 있다. 빨간색 표시를 한 부분처럼 D가 1이어도 E(Enable)이 0이라면 Q가 1을 나타내지 않고 이전의 출력값인 Q = 0, NQ = 1을 그대로 유지하는 것을 확인할 수 있다.

Simulation결과를 바탕으로 다음과 같이 진리표를 작성하였다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
| D | E | Q | NQ |
| 0 | 0 | Qprev | NQprev |
| 0 | 1 | 0 | 1 |
| 1 | 0 | Qprev | NQprev |
| 1 | 1 | 1 | 0 |

Qprev와 NQprev는 각각 이전 Q, NQ의 값을 나타낸다.

FPGA동작을 위해 다음과 같은 xdc코드를 사용한다.

|  |
| --- |
| **D\_Latch.xdc** |
| set\_property IOSTANDARD LVCMOS18 [get\_ports {D}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {E}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {Q}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {NQ}]  set\_property PACKAGE\_PIN F15 [get\_ports {Q}]  set\_property PACKAGE\_PIN F13 [get\_ports {NQ}]  set\_property PACKAGE\_PIN J4 [get\_ports {D}]  set\_property PACKAGE\_PIN L3 [get\_ports {E}] |

이 xdc코드를 통해 FPGA를 동작시키면 다음과 같은 결과를 얻을 수 있을 것이다.

FPGA에서 각 스위치와 LED는

SW1=D, SW2=E

LD1=Q, LD2=NQ

를 나타내며

SW1=1, SW2=1일 때 LD1=ON, LD2=OFF일 것이며

SW1=0, SW2=1일 때 LD1=OFF, LD2=ON일 것이다.

SW2=0이라면 SW1을 조작하여도 LD1, LD2는 변하지 않을 것이다.

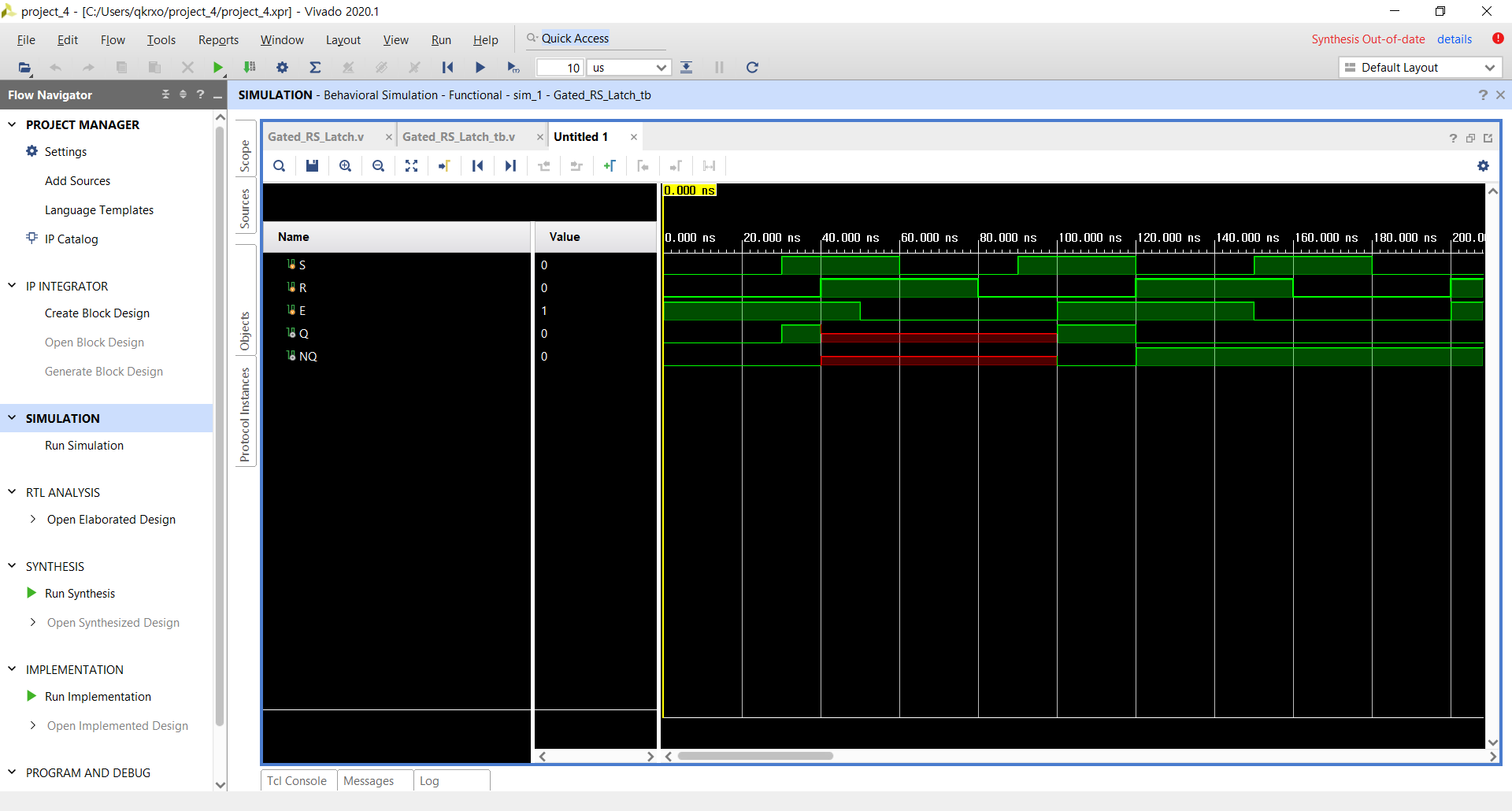
**2. Gated RS Latch의 결과 및 Simulation 과정에 대해서 설명하시오.**

Gated RS Latch를 다음과 같이 Verilog로 구현하였다.

|  |
| --- |
| **Gated\_RS\_Latch** |
| `timescale 1ns / 1ps  module Gated\_RS\_Latch(R, S, E, Q, NQ);  input R,S,E;  output Q, NQ;  reg Q, NQ;    initial begin  Q = 0;  NQ = 0;  end    always @(E or R or S) begin  if(E == 1)  begin  if(S==1 && R==1)  begin  Q <= 1'bx;  NQ <= 1'bx;  end  else if(S==0 && R==0)  begin  Q <= Q;  NQ <= NQ;  end  else  begin  Q <= S;  NQ <= ~S;  end  end  end    endmodule |
| **D\_Latch\_tb** |
| `timescale 1ns / 1ps  module Gated\_RS\_Latch\_tb();  reg S,R,E;  wire Q, NQ;  Gated\_RS\_Latch connect(R, S, E, Q, NQ);  initial begin  S = 0; R = 0;E = 1;  end  always S = #30 ~ S;  always R = #40 ~ R;  always E = #50 ~ E;  endmodule |

입력으로 R, S, E를 받으며 Q와 NQ를 출력한다. 입력 S와 R에 따라 출력값이 바뀌며 E가 0일때는 나머지 두 입력값에 상관없이 이전 출력을 유지하고 E가 1일 때에만 S, R에 따라 출력을 변화시킨다. E가 1일 때 Q는 S와 같은 값을 나타내며 NQ는 Q와 반대되는 값을 나타낸다. E가 1일 때 S와 R이 모두 1이라면 Undefined된 값이 나와야 하므로 두 출력을 모두 X로 설정하였다.

구현한 Verilog코드를 이용하여 Simulation을 하였다.



출력 Q와 NQ가 항상 반대되는 값을 나타내며 E가 0일 때는 이전 출력을 유지, 1일 때는 두 입력 S와R에 따라 출력이 바뀌는 것을 확인할 수 있다. E(Enable)이 1일 때 S=1, R=0이면 Q=1을 나타내며 S=0, R=1일 때 Q=0을 나타낸다. 여기서 세 입력 모두 1이라면 Undefined된 값 X를 나타내는 것 또한 확인할 수 있다.

Simulation결과를 바탕으로 다음과 같이 진리표를 작성하였다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | **Output** | |
| 입력 순서 | R | S | Q | NQ |
| (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 |

위에서부터 아래로 입력 순서에 따라 진리표를 작성하였다.

FPGA동작을 위해 다음과 같은 xdc코드를 사용한다.

|  |
| --- |
| **Gated\_RS\_Latch.xdc** |
| set\_property IOSTANDARD LVCMOS18 [get\_ports {S}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {R}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {E}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {Q}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {NQ}]  set\_property PACKAGE\_PIN F15 [get\_ports {Q}]  set\_property PACKAGE\_PIN F13 [get\_ports {NQ}]  set\_property PACKAGE\_PIN J4 [get\_ports {S}]  set\_property PACKAGE\_PIN L3 [get\_ports {R}]  set\_property PACKAGE\_PIN K3 [get\_ports {E}] |

이 xdc코드를 통해 FPGA를 동작시키면 다음과 같은 결과를 얻을 수 있을 것이다.

FPGA에서 각 스위치와 LED는

SW1=S, SW2=R, SW3=E

LD1=Q, LD2=NQ

를 나타내며

SW1=1, SW2=0, SW3=1일 때 LD1=ON, LD2=OFF일 것이며

SW1=0, SW2=1, SW3=1일 때 LD1=OFF, LD2=ON일 것이다.

SW3=0이라면 SW1을 조작하여도 LD1, LD2는 변하지 않을 것이다.

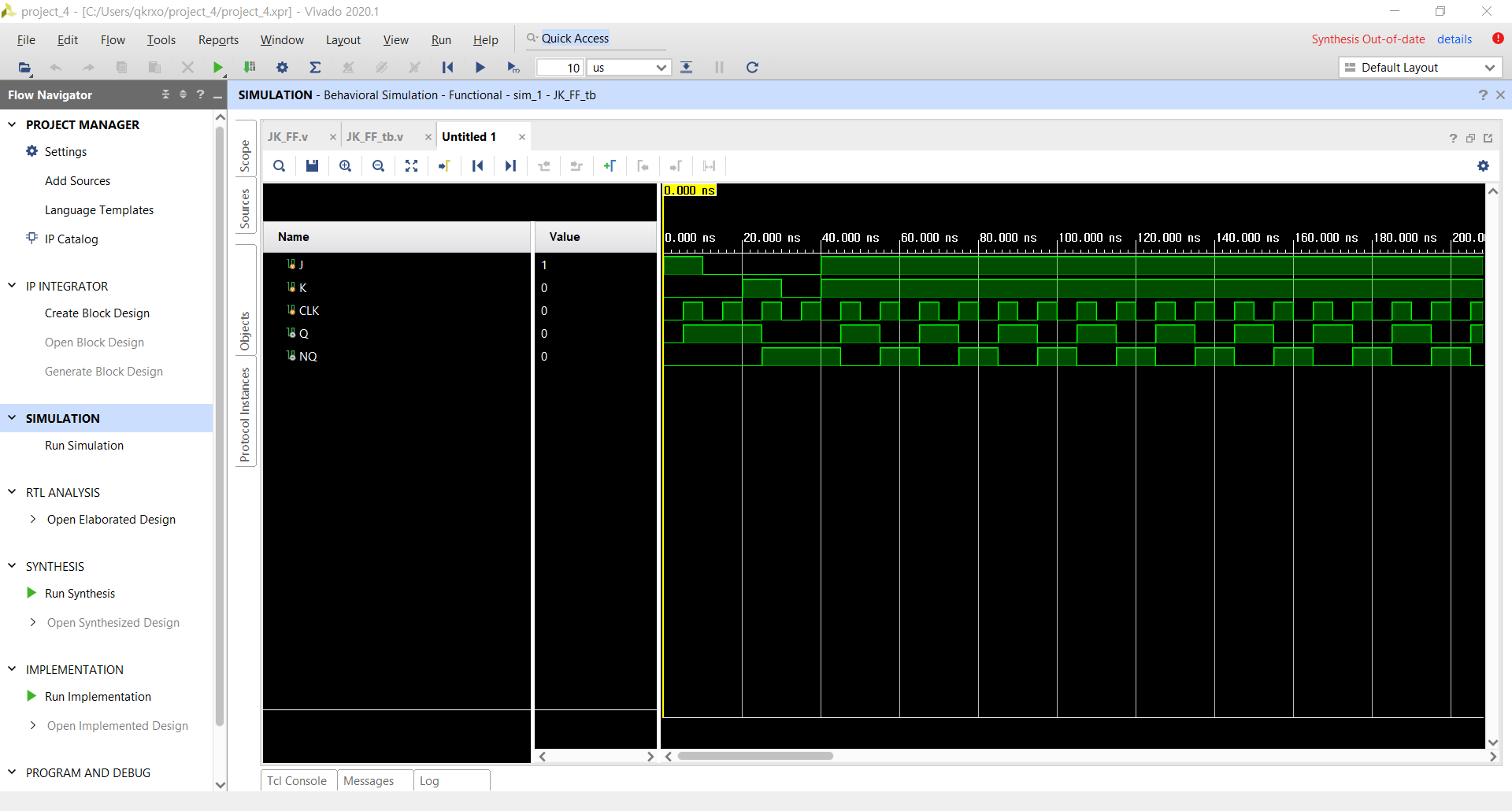
**3. JK Flip-Flop의 결과 및 Simulation과정에 대해서 설명하시오.**

JK Flip-Flop을 다음과 같이 Verilog로 구현하였다.

|  |
| --- |
| **JK\_FF** |
| `timescale 1ns / 1ps  module JK\_FF(CLK, J, K, Q, NQ);  input CLK, J, K;  output Q, NQ;  reg Q, NQ;    initial begin  Q = 0;  NQ = 0;  end    always @(posedge CLK) begin  if((J==1)&&(K==1))  begin  Q <= ~Q;  NQ <= ~NQ;  end  else if((J==1)&&(K==0))  begin  Q <= J;  NQ <= ~J;  end  else if((J==0)&&(K==1))  begin  Q <= ~K;  NQ <= K;  end  else  begin  Q <= Q;  NQ <= NQ;  end  end    endmodule |
| **JK\_FF\_tb** |
| `timescale 1ns / 1ps  module JK\_FF\_tb();  reg J,K,CLK;  wire Q,NQ;  JK\_FF connect(CLK,J,K,Q,NQ);  initial begin  CLK=0; J=1; K=0;  J = #10 0;  K = #10 1;  K = #10 0;  J = #10 1; K = 1;  end;  always CLK = #5 ~CLK;  endmodule; |

입력으로 clock인 CLK와 J, K를 받으며 Q와 NQ를 출력한다. CLK가 posedege, learning edge일 때 J와 K에 따라 출력값이 정해진다. CLK이 1을 나타낼 때 J=1, K=0이면 Q=1, NQ=0을 나타내며 J=0, K=1일 때는 Q=0, NQ=1을 나타낸다. J와 K모두 0이면 이전의 출력값을 유지하며 CLK가 1일 때 J와 K모두 1을 나타낸다면 출력을 반전시킨다.

구현한 Verilog코드를 이용하여 Simulation을 하였다.



출력 Q와 NQ가 항상 반대되는 값을 나타내며 CLK=1일 때에만 입력에 따라 출력이 변화하는 것을 확인할 수 있다. 시뮬레이션 결과에 표시된 부분에서 J와 K가 모두 1이며 CLK가 1을 나타낼 때 마다 출력이 반전되는 것을 알 수 있다.

Simulation결과를 바탕으로 다음과 같이 진리표를 작성하였다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | **Output** | |
| 입력 순서 | J | K | Q | NQ |
| (1) | 0 | 1 | 0 | 1 |
| (2) | 0 | 0 | 0 | 0 |
| (3) | 1 | 0 | 1 | 0 |
| (4) | 0 | 0 | 1 | 0 |
| (5) | 1 | 0 | 1 | 0 |
| (6) | 1 | 1 | 0 | 1 |

위에서부터 아래로 입력 순서에 따라 진리표를 작성하였다.

FPGA동작을 위해 다음과 같은 xdc코드를 사용한다.

|  |
| --- |
| **JK\_FF.xdc** |
| set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets {CLK\_IBUF}]  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 {Q}]  set\_property IOSTANDARD LVCMOS18 [get\_ports {NQ}]  set\_property PACKAGE\_PIN F15 [get\_ports {Q}]  set\_property PACKAGE\_PIN F13 [get\_ports {NQ}]  set\_property PACKAGE\_PIN J4 [get\_ports {J}]  set\_property PACKAGE\_PIN L3 [get\_ports {K}]  set\_property PACKAGE\_PIN K3 [get\_ports {CLK}] |

이 xdc코드를 통해 FPGA를 동작시키면 다음과 같은 결과를 얻을 수 있을 것이다.

FPGA에서 각 스위치와 LED는

SW1=J, SW2=K, SW3=CLK

LD1=Q, LD2=NQ

를 나타내며

SW3=0이라면 SW1을 조작하여도 LD1, LD2는 변하지 않을 것이다.

SW1=1, SW2=0, SW3=1일 때 LD1=ON, LD2=OFF일 것이며

SW1=0, SW2=1, SW3=1일 때 LD1=OFF, LD2=ON일 것이다.

마지막으로 SW1=1, SW2=1 일 때 SW3(CLK)스위치를 ON OFF반복하면 LD1과 LD2가 계속 반전된 값으로 바뀔 것이다.

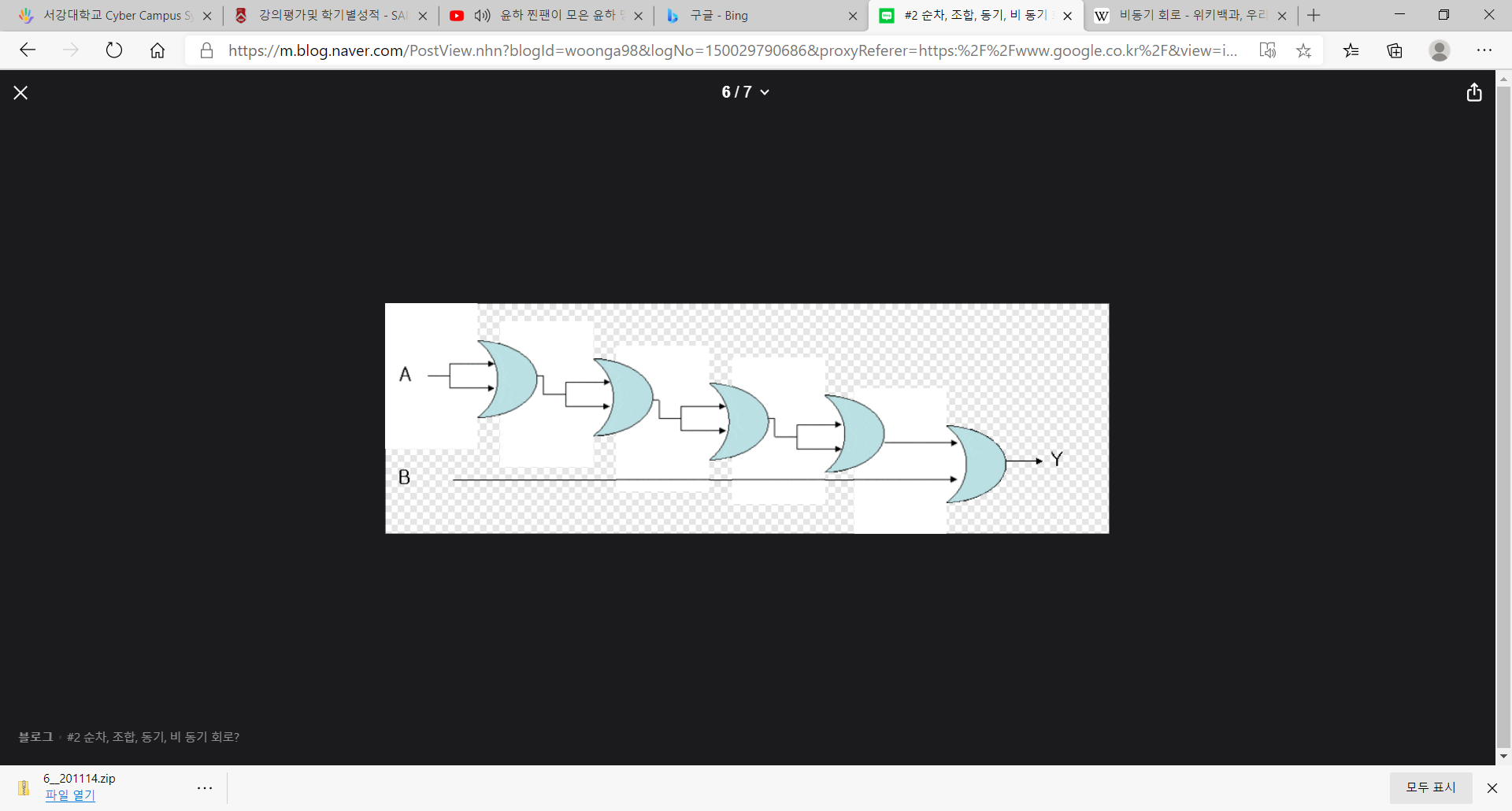
**4. 결과 검토 및 논의 사항.**

D Latch, RS Latch, JK Flip-Flop를 Verilog로 구현하였으며 Simulation을 통해 결과를 확인하였고 이를 토대로 진리표를 작성한 뒤 FPGA동작을 살펴보았다. RS래치에서 두 입력이 모두 1인 경우 X값을 나타내도록 구현하였다. 이 때, D래치에서는 RS래치에서 입력 R과 S가 D와 D’로 표현되기 때문에 두 입력이 모두 1인 경우가 존재할 수 없게 되어 정상적으로 동작할 수 있음을 확인하였다. 두 종류의 래치와 달리 JK Flip-Flop에서는 Clock(=CLK)가 존재하도록 구현하였다. 또한 JK플립플롭은 두 입력이 모두 1일 때 X값을 나타내는 것이 아닌 출력을 반전시키는 것으로 구현하였다.

D, RS래치와 JK플립플롭 이외에 T플립플롭이나 Master-Slave플립플롭 또한 알아보면 좋을 거 같다.

**5. 추가 이론 조사 및 작성.**

동기 회로(Synchronous Circuit)란 Clock신호에 의해 동기화가 되는 회로이다. 반대 개념으로 비동기 회로(Asynchronous Circuit)가 존재한다. 이번 실습에서 살펴본 JK Flip-Flop이 동기 회로 중 하나이다. 조합논리회로는 모두 비동기 회로이며 순차논리회로에서 동기 회로와 비동기 회로로 구분이 된다. 동기 회로는 다음과 같은 이유 때문에 사용된다.



위와 같은 회로가 있다고 가정하자. A와 B 둘 중 하나가 1이라면 출력 Y가 1이 된다. 두 입력이 모두 1이였다가 모두 0으로 변경했을 시, 지연시간 때문에 Y가 바로 그 순간에 0으로 변경되지 않는다. 예를 들어 위의 회로에서 OR게이트의 지연시간이 0.1초라고 가정하면 출력 Y가 입력 A, B를 반영하는데 0.5초가 걸리게 되는 것이다.

스크린샷, 측정기이(가) 표시된 사진

자동 생성된 설명

따라서 위와 같은 동기 회로를 사용한다. 순차 논리 회로에서 A와 B에 의해 저장된 값을 Clk에 따라 다음 저장소에 보낼 수 있다. 즉, Clk의 주기를 조절하고 적절한 위치에 저장소를 두게 되면 먼저 입력된 값들이 출력에 반영되기 전 다음 입력값을 넣어줄 수 있다. 이를 활용하여 빠른 속도로 처리를 할 수 있다.