**컴퓨터공학 설계 및 실험Ⅱ**

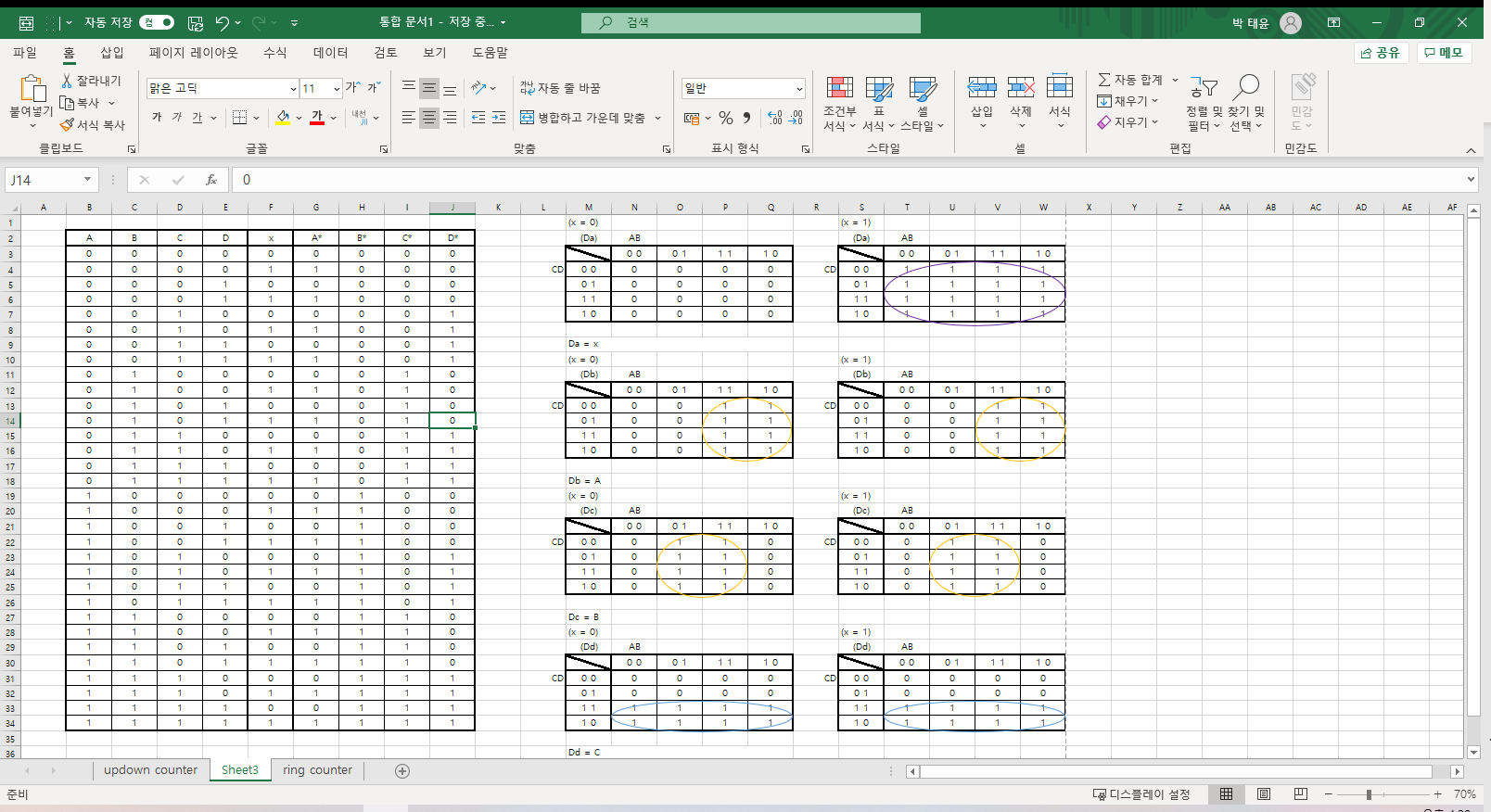
13주차 결과보고서

서강대학교 공학부 컴퓨터공학 전공

20171646 박태윤

**1. 4-bit Shift Register의 결과 및 Simulation과정에 대해서 설명하시오.**

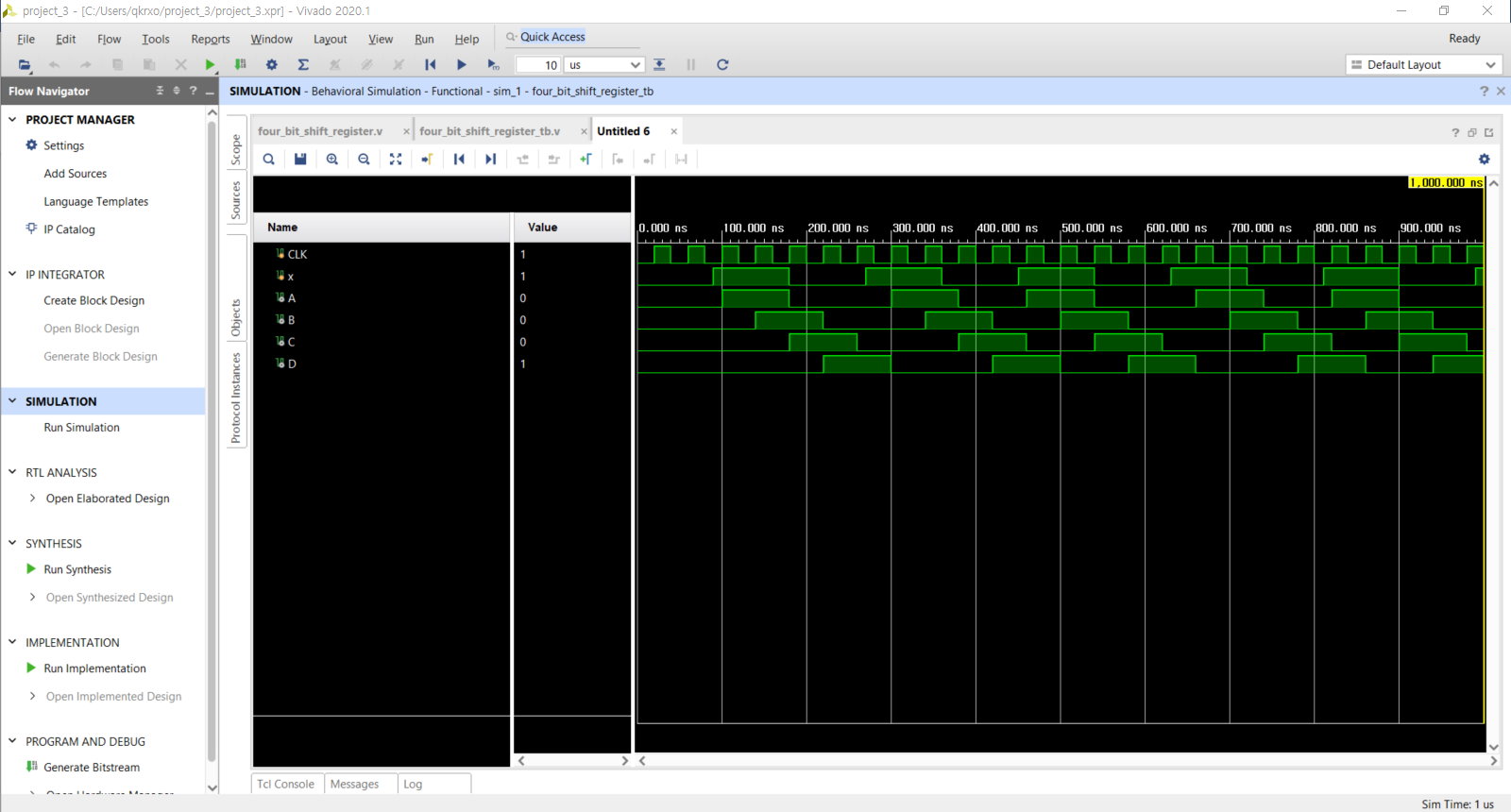
다음 그림과 같이 상태도를 작성하여 카르노 맵을 통해 각 플립플롭의 입력에 해당하는 식을 구하였다. D플립플롭을 사용하였는데 상태도에서 각 플립플롭의 입력 Da, Db, Dc, Dd는 입력에 대한 다음 상태인 A\*, B\*, C\*, D\*와 같다.



4-bit Shift Register를 다음과 같이 구현하였다.

|  |
| --- |
| **four\_bit\_shift\_register** |
| `timescale 1ns / 1ps  module four\_bit\_shift\_register(CLK,A,B,C,D,x);  input CLK, x;  output A,B,C,D;  assign Da=x;  assign Db=A;  assign Dc=B;  assign Dd=C;  D\_FF d1(A,Da,CLK);  D\_FF d2(B,Db,CLK);  D\_FF d3(C,Dc,CLK);  D\_FF d4(D,Dd,CLK);  endmodule  module D\_FF(Q,Dx,CLK);  input Dx,CLK;  output Q;  reg Q,NQ;  initial begin  Q=0;  NQ=0;  end  always @(posedge CLK) begin  Q<=Dx;  NQ<=~Dx;  end  endmodule |
| **four\_bit\_shift\_register\_tb** |
| `timescale 1ns / 1ps  module four\_bit\_shift\_register\_tb;  reg CLK, x;  wire A,B,C,D;  four\_bit\_shift\_register connect(CLK,A,B,C,D,x);  initial begin  CLK=1'b0;  x=1'b0;  end  always CLK = #20 ~CLK;  always x = #90 ~x;  initial begin  #1000  $finish;  end  endmodule |

다음은 코드를 토대로 Simulation을 한 결과이다.





CLK(=clock)이 1을 나타낼 때 마다 우측으로 하나씩 밀려나는 것과 입력 x가 1을 나타낼 때 A에 1이 추가되는 것을 확인할 수 있다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number  Clock Transitions | IN(x) | A | B | C | D |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 |
| 4 | 1 | 1 | 0 | 0 | 1 |
| 5 | 1 | 1 | 1 | 0 | 0 |
| 6 | 0 | 0 | 1 | 1 | 0 |
| 7 | 0 | 0 | 0 | 1 | 1 |

위의 표는 시뮬레이션 결과를 토대로 작성한 Output Table이다.

FPGA동작을 살펴보면

SW1=x, SW2=CLK

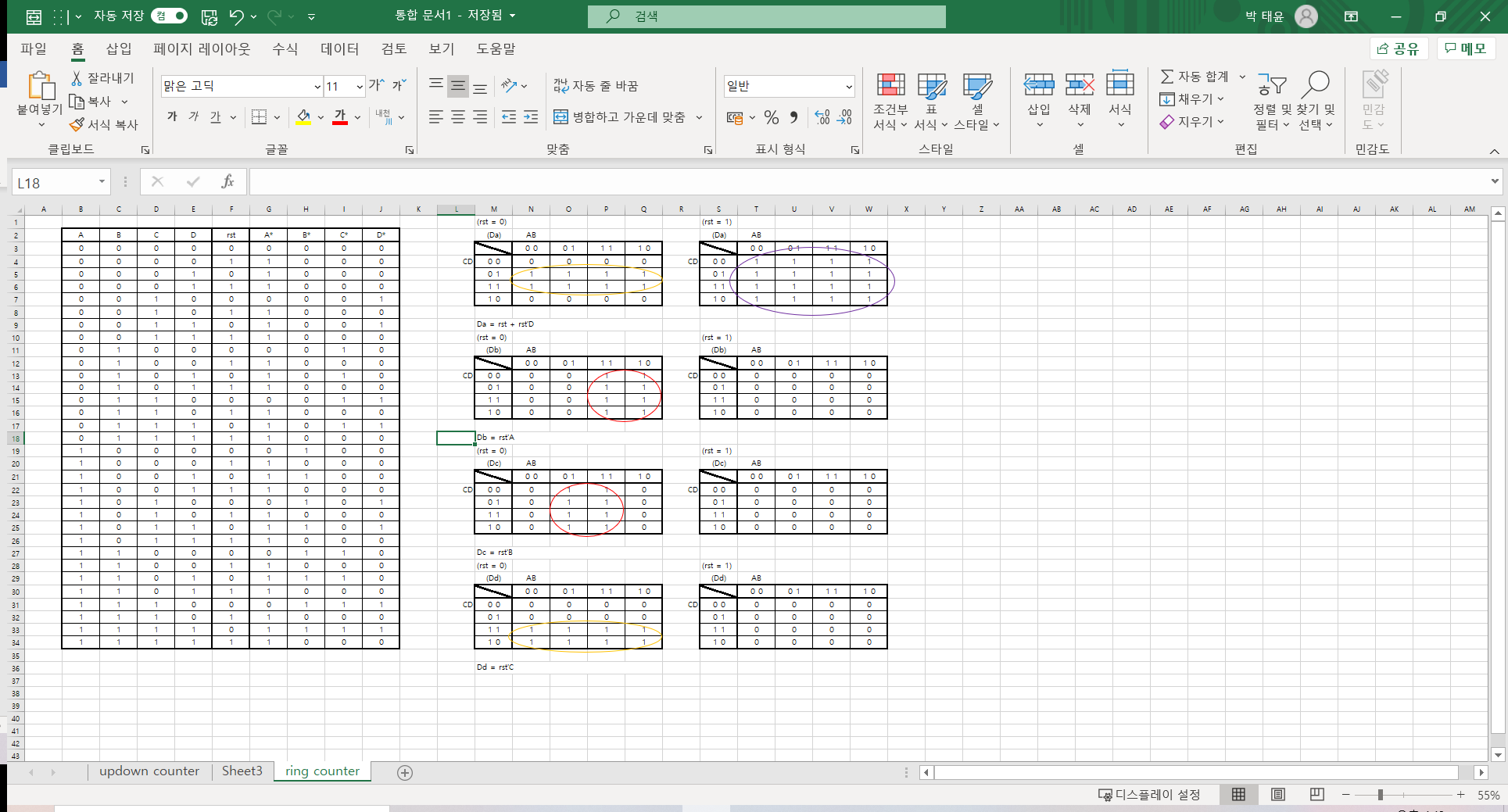
LD1=A, LD2=B, LD3=C, LD4=D

라 가정했을 때

SW1=ON과 SW2=ON을 통해 LD1=ON, LD2=OFF, LD3=OFF, LD4=OFF를 나타내고 SW2=ON&OFF을 통해 (LD1, LD2, LD3, LD4) = (ON, OFF, OFF, OFF) -> (OFF, ON, OFF, OFF) -> (OFF, OFF, ON, OFF) -> (OFF, OFF, OFF, ON)를 나타내는 것을 확인할 수 있을 것이다.

**2. 4-bit Ring counter의 결과 및 Simulation과정에 대해서 설명하시오.**

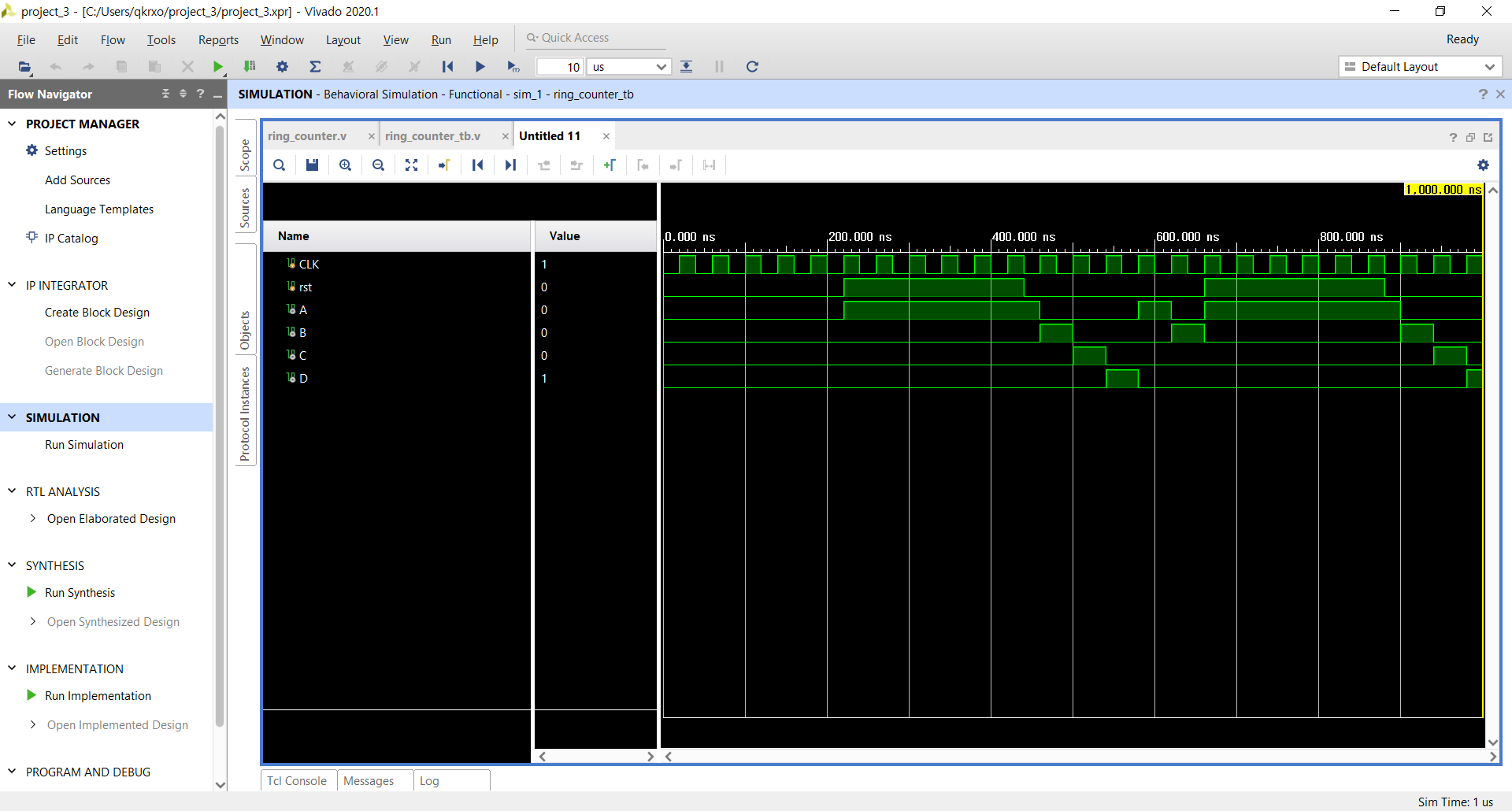
다음 그림과 같이 상태도를 작성하여 카르노 맵을 통해 각 플립플롭의 입력에 해당하는 식을 구하였다.



4-bit Ring Counter를 다음과 같이 구현하였다.

|  |
| --- |
| **ring\_counter** |
| `timescale 1ns / 1ps  module ring\_counter(CLK,A,B,C,D,rst);  input CLK, rst;  output A,B,C,D;  assign Da= rst | (~rst)&D;  assign Db=(~rst)&A;  assign Dc=(~rst)&B;  assign Dd=(~rst)&C;  D\_FF d1(A,Da,CLK);  D\_FF d2(B,Db,CLK);  D\_FF d3(C,Dc,CLK);  D\_FF d4(D,Dd,CLK);  endmodule  module D\_FF(Q,Dx,CLK);  input Dx,CLK;  output Q;  reg Q,NQ;  initial begin  Q=0;  NQ=0;  end  always @(posedge CLK) begin  Q<=Dx;  NQ<=~Dx;  end  endmodule |
| **ring\_counter\_tb** |
| `timescale 1ns / 1ps  module ring\_counter\_tb;  reg CLK,rst;  wire A,B,C,D;  ring\_counter connect(CLK,A,B,C,D,rst);  initial begin  CLK=1'b0;  rst=1'b0;  end  always CLK = #20 ~CLK;  always rst = #220 ~rst;  initial begin  #1000  $finish;  end  endmodule |

다음은 코드를 토대로 Simulation을 한 결과이다.





CLK(=clock)이 1을 나타낼 때 마다 우측으로 하나씩 밀려나다가 D=1을 나타날 때 CLK=1이 들어오면 맨 마지막 플립플롭의 출력이 첫번째 플립플롭의 입력에 연결되어 A가 1을 나타내는 것을 확인할 수 있다. rst(=reset)이 1을 나타내는 경우 초기값인 1000을 나타낸다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number  Clock Transitions | rst | A | B | C | D |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 0 | 0 |
| 6 | 0 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 0 | 1 | 0 |

위의 표는 시뮬레이션 결과를 토대로 작성한 Output Table이다.

FPGA동작을 살펴보면

SW1=rst, SW2=CLK

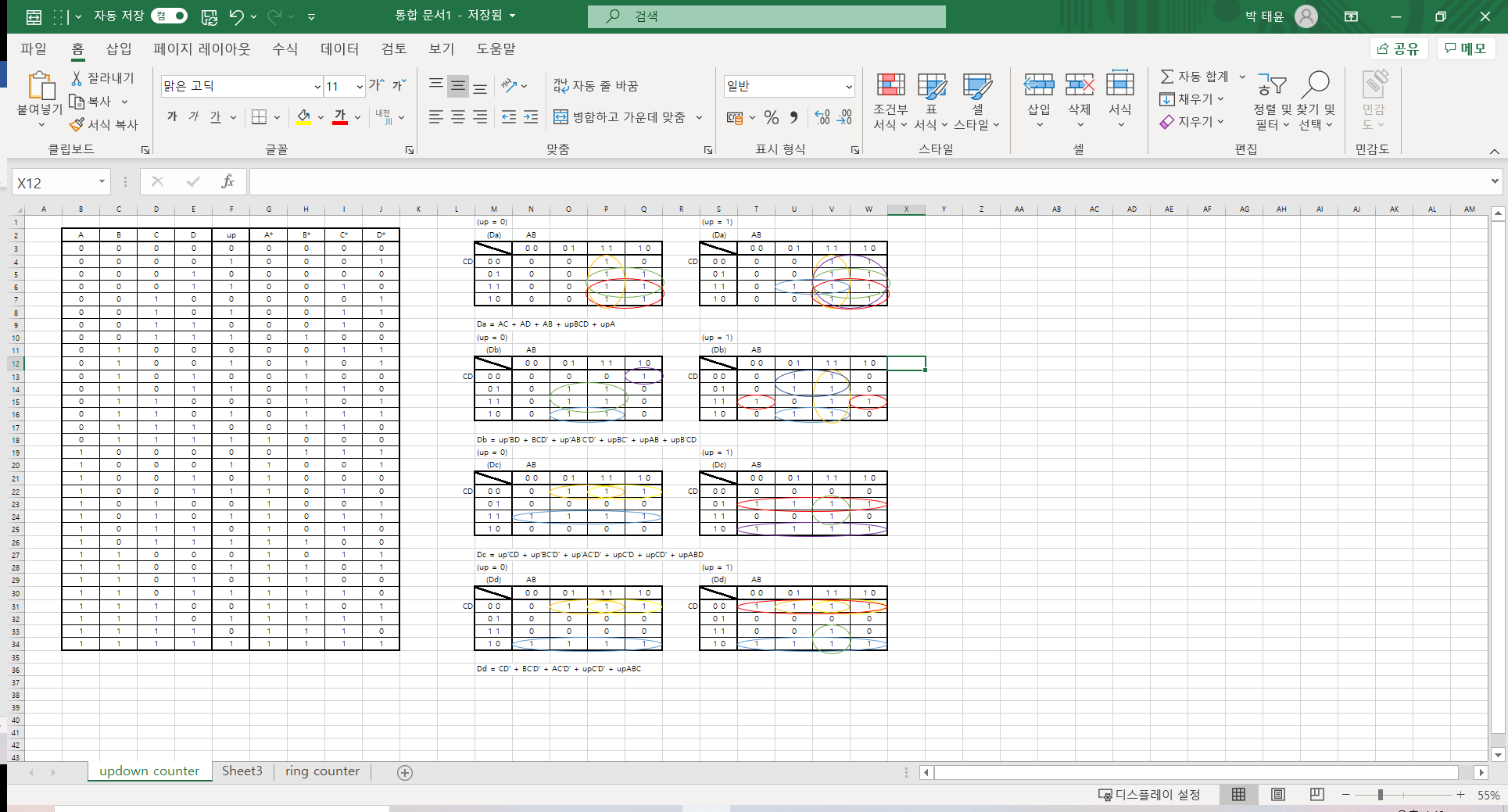
LD1=A, LD2=B, LD3=C, LD4=D

라 가정했을 때

SW1=ON과 SW2=ON을 통해 LD1=ON, LD2=OFF, LD3=OFF, LD4=OFF를 나타내고 SW2=ON&OFF을 통해 (LD1, LD2, LD3, LD4) = (ON, OFF, OFF, OFF) -> (OFF, ON, OFF, OFF) -> (OFF, OFF, ON, OFF) -> (OFF, OFF, OFF, ON) -> (ON, OFF, OFF, OFF)…를 나타내는 것을 확인할 수 있을 것이다.

**3. 4-bit Up/Down counter의 결과 및 Simulation과정에 대해서 설명하시오.**

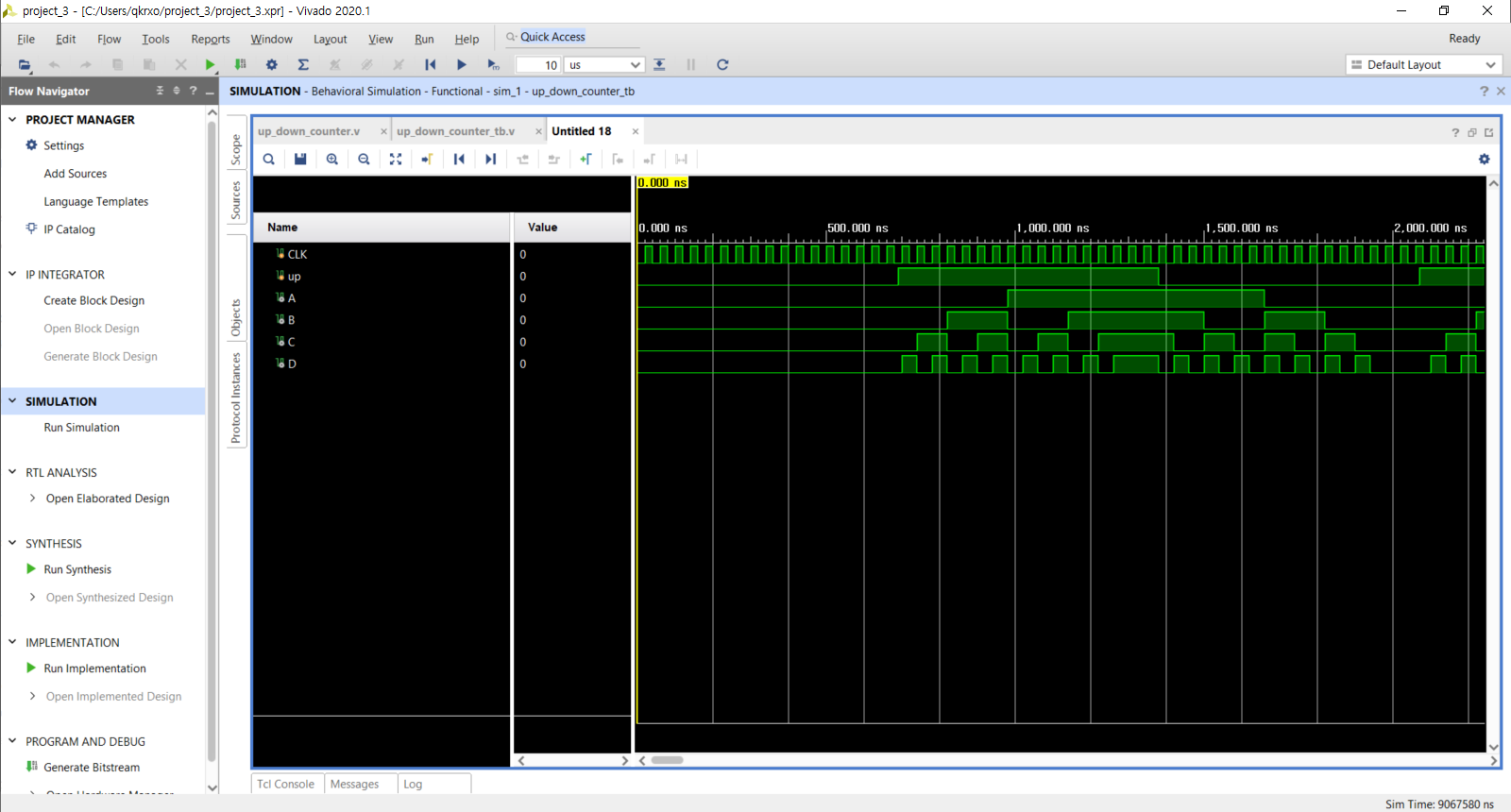
다음 그림과 같이 상태도를 작성하여 카르노 맵을 통해 각 플립플롭의 입력에 해당하는 식을 구하였다.



4-bit Up/Down Counter를 다음과 같이 구현하였다.

|  |
| --- |
| **up\_down\_counter** |
| `timescale 1ns / 1ps  module up\_down\_counter(CLK,A,B,C,D,up);  input CLK, up;  output A,B,C,D;  assign Da=A&C | A&D | A&B | up&B&C&D | up&A;  assign Db=(~up)&B&D|B&C&(~D)|(~up)&A&(~B)&(~C)&(~D)|up&A&B|up&B&(~C)|up&(~B)&C&D;  assign Dc=(~up)&C&D|(~up)&B&(~C)&(~D)|(~up)&A&(~C)&(~D)|up&D&(~C)|up&C&(~D)|up&A&B&D;  assign Dd=C&(~D)|B&(~C)&(~D)|A&(~C)&(~D)|up&(~C)&(~D)|up&A&B&C;  D\_FF d1(A,Da,CLK);  D\_FF d2(B,Db,CLK);  D\_FF d3(C,Dc,CLK);  D\_FF d4(D,Dd,CLK);  endmodule  module D\_FF(Q,Dx,CLK);  input Dx,CLK;  output Q;  reg Q,NQ;  initial begin  Q=0;  NQ=0;  end  always @(posedge CLK) begin  Q<=Dx;  NQ<=~Dx;  end  endmodule |
| **up\_down\_counter\_tb** |
| `timescale 1ns / 1ps  module up\_down\_counter\_tb;  reg CLK, up;  wire A,B,C,D;  up\_down\_counter connect(CLK,A,B,C,D,up);  initial begin  CLK=1'b0;  up=1'b0;  end  always CLK = #20 ~CLK;  always up = #690 ~up;  initial begin  #1000  $finish;  end  endmodule |

다음은 코드를 토대로 Simulation을 한 결과이다.



CLK(=clock)이 1을 나타낼 때 입력 up이 1을 나타낸다면 0001->0010->0011…과 같이 4-bit출력이 1씩 증가하고 up이 0을 나타낸다면 1111->1110->1101…과 같이 출력이 1씩 감소하는 것을 확인할 수 있다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number  Clock Transitions | up | A | B | C | D |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 | 1 |
| 4 | 1 | 0 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 | 0 | 1 |
| 6 | 1 | 0 | 1 | 1 | 0 |
| 7 | 1 | 0 | 1 | 1 | 1 |
| 8 | 1 | 1 | 0 | 0 | 0 |
| 9 | 1 | 1 | 0 | 0 | 1 |
| 10 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 1 | 1 |
| 12 | 0 | 0 | 1 | 1 | 0 |
| 13 | 0 | 0 | 1 | 0 | 1 |
| 14 | 0 | 0 | 1 | 0 | 0 |

위의 표는 시뮬레이션 결과를 토대로 작성한 Output Table이다.

FPGA동작을 살펴보면

SW1=up, SW2=CLK

LD1=A, LD2=B, LD3=C, LD4=D

라 가정했을 때

처음에 SW1=ON과 SW2=ON을 통해 LD1=OFF, LD2=OFF, LD3=OFF, LD4=ON를 나타내고 SW2=ON&OFF을 통해 (LD1, LD2, LD3, LD4) = (OFF, OFF, OFF, ON) -> (OFF, OFF, ON, OFF) -> (OFF, OFF, ON, ON) -> (OFF, ON, OFF, OFF) -> (OFF, ON, OFF, ON)->…..->(ON, ON, ON, ON)과 같이 1씩 증가하다가 SW1를 내리고 SW2=ON&OFF동작을 하면 (LD1, LD2, LD3, LD4) = (ON, ON, ON, ON) -> (ON, ON, ON, OFF) -> (ON, ON, OFF, ON) ->…..->(OFF, OFF, OFF, OFF)와 같이 1씩 감소하는 것을 확인할 수 있을 것이다.

**4. 결과 검토 및 논의 사항.**

4-bit shift register, ring counter, up down counter의 상태도를 작성한 뒤 이를 토대로 각 플립플롭의 입력들에 대한 식을 구하였으며 Verilog로 구현한 뒤 Simulation결과를 살펴보았다. 3가지 회로에 대해 모두 D플립플롭을 사용하였는데, 현재 상태 (ABCD)에서 입력에 대한 다음 상태인 (A\*B\*C\*D\*)가 각각 한 비트에 대한 플립플롭의 입력인 (Da, Db, Dc, Dd)와 같은 것을 확인하였다.

추가적으로 shift register와 ring counter는 비슷한 동작을 하는데, ring counter는 shift register와 달리 마지막 플립플롭의 출력이 첫번째 플립플롭의 입력으로 들어가기 때문에 D가 1을 나타낼 때 CLK가 1을 나타낸다면 A의 값이 1로 바뀌는 것을 확인하였다. 이는 rst(=reset)신호를 활용하여 Da= rst | (~rst)&D와 같이 구현하였다. rst가 1일 때는 1000을 출력하고 rst가 0일때는 Da가 D를 나타낸다.

**5. 추가 이론 조사 및 작성.**

주파수 카운터란 특정한 설정 시간동안의 전기적 진동수를 측정하기 위한 측정기이다. 보통 1초를 기준으로 진동수를 카운트하며 이를 Hertz(Hz)로 표시한다. 측정 정밀도가 높으며 조작이 비교적 쉬운 특징을 가지고 있다.

