13주차 결과보고서

전공: 심리학과 학년: 3학년 학번: 20190345 이름: 김동현

1.

4비트 시프트 레지스터는 4개의 플립플롭이 직렬로 연결되어 플립플롭의 출력이 다음 플립플롭의 입력으로 구성된다. 본 실험에서는 플립플롭을 구현하는 대신, 변수에 값을 할당하는 것으로 상태를 저장하였다. 아래는 4비트 시프트 레지스터를 구현한 베릴로그 코드이다.

테이블이(가) 표시된 사진

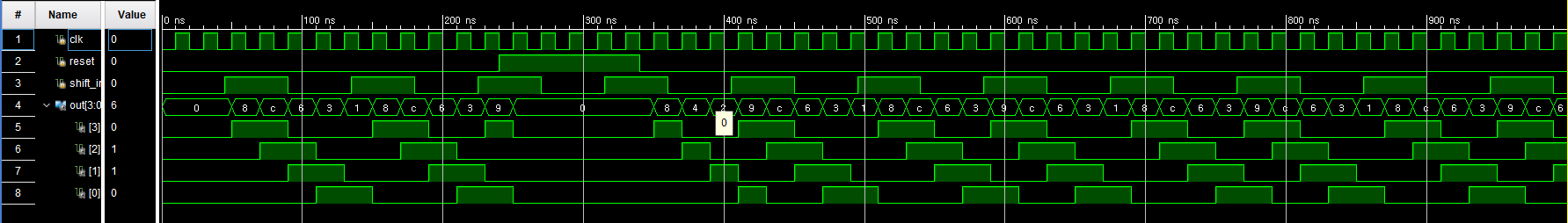
자동 생성된 설명

입력값으로 클럭 펄스를 의미하는 clk, 초기값으로 리셋하기 위한 입력 reset, 레지스터에서 시프트가 일어날 때 입력으로 주어질 값을 의미하는 shift\_in을 선언하였다. 출력값으로는 크기 4의 out 배열을 선언하여 레지스터의 상태를 출력하고자 한다. Initial문을 활용하여 레지스터의 초기값을 0000으로 설정하였다. 이후 always문을 통해 clk가 rising edge인 순간 입력을 받도록 한다. 이때 reset값이 1이면, 0000으로 레지스터를 리셋한다. 그렇지 않은 경우에는 레지스터에 저장된 값을 오른쪽으로 한칸씩 옮긴다. 또한 shift\_in값을 레지스터에 입력한다.

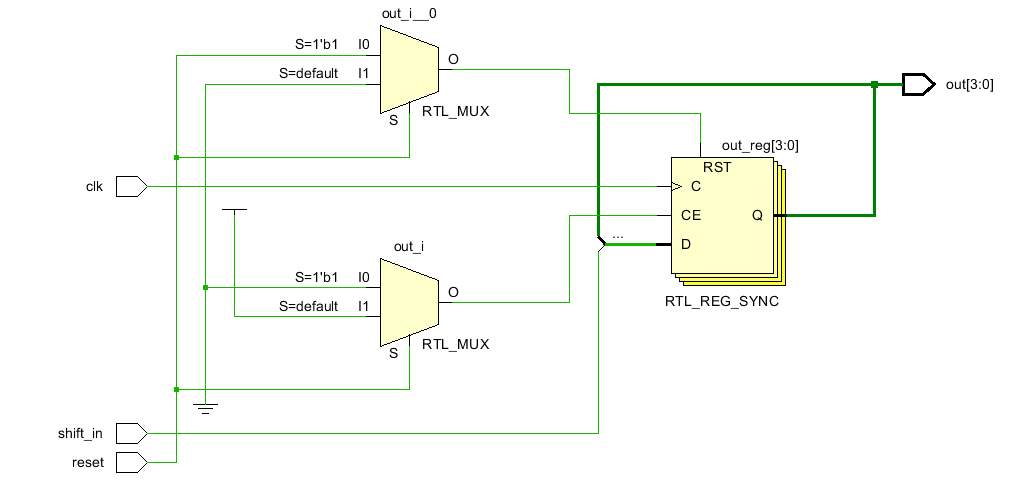
테이블이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 코드는 다음과 같다. 우선 clk, reset, shift\_in 값을 모두 0으로 초기값으로 설정한다. 이후 clk은 10ns마다, shift\_in은 45ns마다 바뀌도록 해, 클럭 펄스의 변화에 따라 shift\_in값이 다양하게 설정될 수 있도록 하였다. 이후 reset의 기능을 확인하기 위해 240ns에 reset을 수행한다.



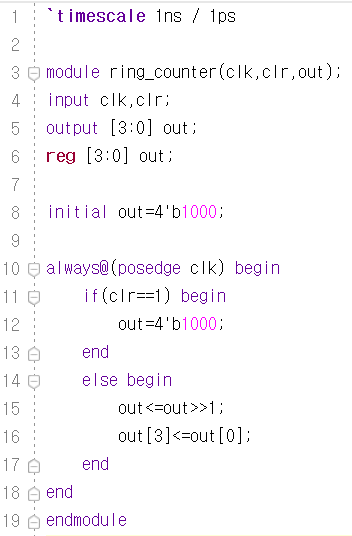
시뮬레이션 결과는 위 그림과 같다. Clk이 rising edge인 순간의 shift\_in, reset값에 따라 레지스터의 상태가 변화하는 것을 알 수 있다. Clk가 rising하는 순간 레지스터의 값이 하나씩 이동하는 것을 확인할 수 있으며, 이때 첫 비트의 값은 shift\_in값이 입력된다. 뿐만 아니라 reset이 1이 된다면, 레지스터가 0000으로 초기화되는 것 역시 확인할 수 있다.



4비트 시프트 레지스터의 schematic은 다음과 같다.

2.

4비트 링 카운터는 4비트 시프트 레지스터를 응용한 카운터이다. 링 카운터의 특징은 레지스터 저장된 마지막 비트가 시프트가 이루어질 때, 사라지는 것이 아닌 링 카운터의 입력으로 주어진다는 점이다. 아래는 4비트 링 카운터를 구현한 베릴로그 코드이다.

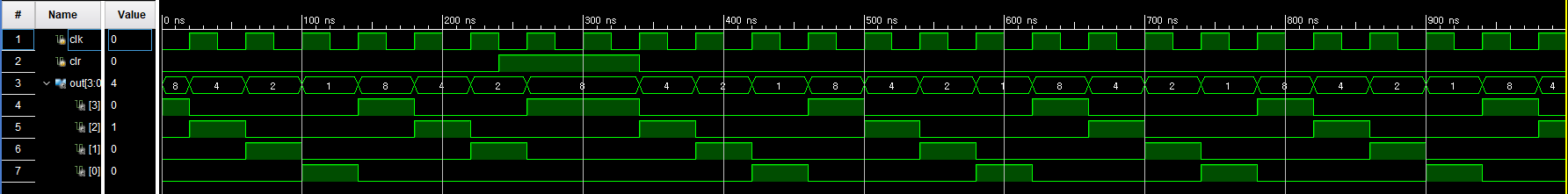


입력값으로 클럭 펄스를 의미하며 링 카운터의 시프트의 실행을 담당하는 clk, 초기값으로 리셋하기 위한 입력 clr가 선언하였다. 출력값으로는 크기 4의 out 배열을 선언하여 레지스터의 상태를 출력하고자 한다. Initial문을 할용하여 레지스터의 초기값을 1000으로 설정하엿다. 이후 always문을 통해 clk가 rising edge인 순간 시프트를 실행하고자 한다. 이때 reset값이 1이면 초값인 1000으로 레지스터를 리셋한다. 그렇지 않은 경우에는 레지스터에 저장된 값을 오른쪽으로 한 칸 씩 옮기며, 레지스터의 마지막 값은 레지스터의 초기값의 입력으로 주어진다.

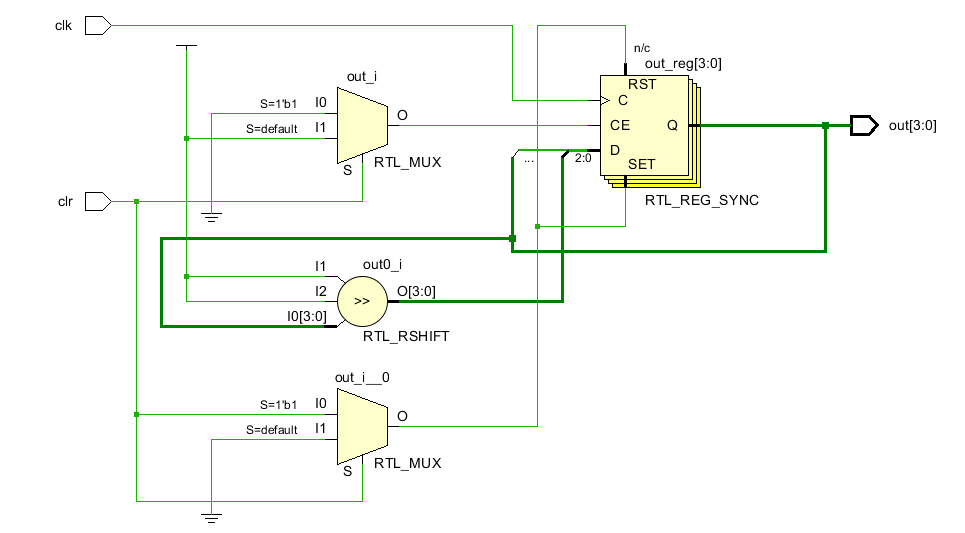
테이블이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 코드는 다음과 같다. 우선 clk, clr값을 모두 0으로 초기화 한다. 이후 clk는 20ns마다 바뀌도록 해, 클럭 펄스의 변화에 따라 링 카운터가 작동하도록 한다. 이후 240ns에서는 clr함수를 1로 변경하여 링 카운터의 리셋 과정 작동 여부를 확인한다.



시뮬레이션 결과는 위 그림과 같다. Clk이 rising edge인 순간 레지스터의 shift가 발생한다. Out[3]의 값이 out[2]로, out[2]의 값이 out[1]로, out[1]의 값이 out[0]으로, out[0]의 값이 out[3]으로 동시에 할당되는 것을 확인할 수 있다. 뿐만 아니라 clr이 1이 된다면, 레지스터가 초기값인 1000으로 초기화되는 것 역시 확인할 수 있다.



4비트 링 카운터의 schematic은 다음과 같다.

3.

4비트 업다운 카운터는 사용자의 업, 다운 여부 선택에 따라 카운터의 작동을 결정한다. 4비트는 총 0(0000)부터 15(1111)까지 표현 가능하다. 이때 사용자가 업 카운터를 선택한다면 오름차순으로, 다운 카운터를 선택한다면 내림차순으로 수를 표현한다. 이때 오름차순일 경우 1111 다음 수는 0000이며, 내림차순일 경우 0000 다음 수는 1111이다. 또한 업, 다운 여부를 확인하기 위해 7 segment display에 u(up), d(down)을 나타낸다. 아래는 4비트 업다운 카운터를 구현한 베릴로그 코드이다.

테이블이(가) 표시된 사진

자동 생성된 설명

입력값으로 카운터의 카운팅 방향을 설정할 up\_down, 클럭 펄스를 의미하는 clk, 초기값으로 리셋하기 위한 clr을 선언하였다. 출력값으로는 카운터의 상태를 나타낼 크기 4의 배열 out과 7 segment display를 표현하기 위한 크기 7의 배열 seg와 dp를 선언하였다. 7 segment display의 출력 여부를 나타내는 dp변수를 1로 설정하였다. 이후 initial문으로 활용하여 초기값으로 out은 0000, seg는 0000000으로 설정하여 FPGA에서 처음에 아무런 불이 들어오지 않도록 했다. 이후 always문을 통해 clk가 rising edge인 순간 업/다운 여부의 결정 및 카운팅을 실행하도록 했다. Clr값이 1인 경우는 리셋을 수행하기 위해 out=0000, seg=0000000으로 설정한다. 그렇지 않다면, up\_down값에 따라 카운팅 방향을 설정한다. Up\_down==1인 경우 오름차순으로 수를 세기 때문에 out값에 1을 더한다. 이후 7 segment display에 u를 표현하기 위해 seg값을 0111110으로 설정한다. Up\_down==0인 경우 내람차순으로 수를 세기 때문에 out값에 1을 뺀다. 이후 7 segment display에 d를 표현하기 위해 seg값을 1011110으로 설정한다.

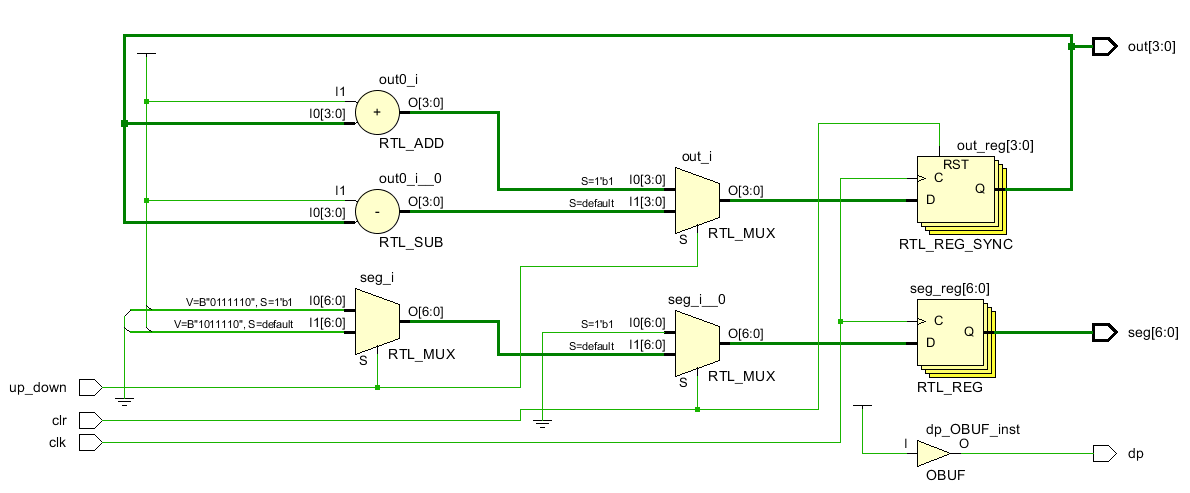
테이블이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 코드는 다음과 같다. 우선 clk, clr값을 0으로 초기화한다. 이후 clk는 10ns마다 바뀌도록 해, 클럭 펄스의 변화에 따라 업다운 카운터가 작동하도록 한다. 이후 350ns이후 down counter에서의 리셋 과정 작동 여부를 확인한다. 100ns이후 up\_down값을 1로 바꾸어 up counter의 작동여부를 확인한다. 350ns이후 up counter에서의 리셋 과정 작동 여부를 확인한다.



시뮬레이션 결과는 위 그림과 같다. Clk가 rising edge인 순간 카운터의 상태 변화가 발생한다. Up\_down변수가 1인 경우 오름차순으로 카운터가 작동하고, 0인 경우 내림차순으로 카운터가 작동하는 것을 확인할 수 있다. Clr값이 1인 경우 카운터가 0000으로 초기화 되는 것을 업카운터와 다운 카운터 모두 확인할 수 있다. 또한 업카운터인 경우 1111 다음 0000으로 값이 연결되어 있는 것과 다운 카운터인 경우 0000다음 1111으로 값이 연결되어 있는 것을 확인할 수 있다. 7 segment display를 나타내기 위해 dp의 값이 1로 유지되고 있으며, 초기화 상태에서는 0000000으로 업인 경우 0111110으로 u를 표시하며, 다운인 경우 1011110으로 d를 표시하고자 함을 알 수 있다.



4비트 업 다운 카운터의 schematic은 다음과 같다.

4.

시프트 레지스터, 링 카운터, 업다운 카운터를 베릴로그를 통해 구현하였고, 시뮬레이션과 FPGA의 비교를 통해 동작을 검증하였다. 레지스터의 구현은 베릴로그 내에서 배열로 나타내었다. 뿐만 아니라 데이터의 이동 과정에서 데이터의 손실을 막기 위해 non-blocking statement(<=)을 사용하였다. 모든 레지스터와 카운터는 clk(클록 펄스)가 rising edge인 순간 작동하도록 하였으며, 그렇지 않은 경우에는 값의 변화에 상태가 변화하지 않았다. 시프트 레지스터는 입력으로 들어온 값이 오른쪽으로 밀려가며 데이터가 저장되는 레지스터이다. 링 카운터는 시프트 레지스터와의 작동과정이 동일하지만, 새로운 입력이 새롭게 들어오는 것이 아니라 레지스터의 마지막 데이터 값을 입력으로 활용한다는 점이 시프트 레지스터와 링 카운터의 차이점이다. 업 다운 카운터는 사용자가 업 카운터, 다운 카운터를 선택할 수 있도록 하였으며, 이에 따라 0000부터 1111까지 카운터의 기능을 수행한다. 이때 0000과 1111사이를 연결하여 카운터가 중간에 종료되지 않도록 하였으며, 7 segment display를 활용하여 업 카운터와 다운 카운터 중 어느 카운터가 실행되고 있는지 알파벳으로 표현하였다.

5.

예비보고서에서 조사한 johnson couter를 베릴로그를 통해 구현한다면 다음과 같다. 우선 johnson counter의 회로도를 살펴본다면 레지스터의 마지막 출력값의 complement값이 레지스터의 입력값으로 주어지는 것을 확인할 수 있다. 이를 바탕으로 디자인 소스 코드와 시뮬레이션 코드를 구현하면 다음과 같다.

테이블이(가) 표시된 사진

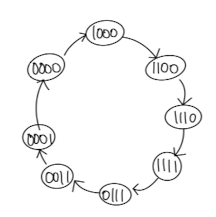
자동 생성된 설명

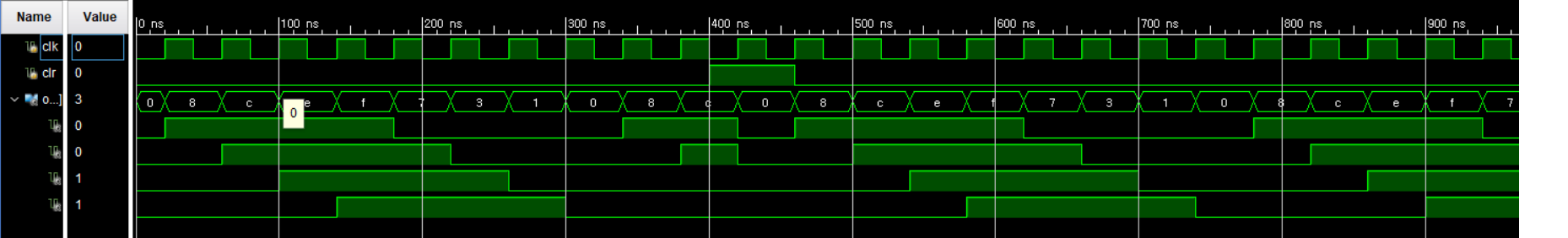
링 카운터와 다르게 마지막 입력의 complement가 레지스터의 첫 입력으로 들어간 것을 확인할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

시뮬레이션 결과를 살펴본다면, 아래와 같은 state diagram의 순서로 counter가 실행되는 것을 확인할 수 있다.



  
state diagram과 같이 상태가 변화하는 것을 확인할 수 있으며, clr변수가 1일때, 0000으로 카운터가 리셋되는 것 역시 확인할 수 있다.