12주차 결과보고서

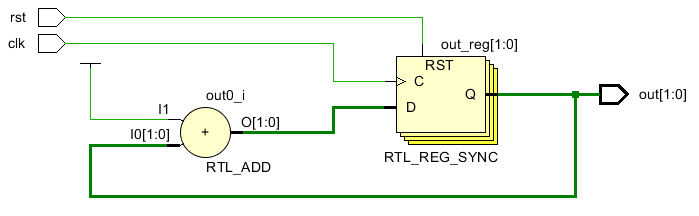
전공: 컴퓨터공학과 학년: 2학년 학번: 20191619 이름: 이동석

**1. 2-bit counter**

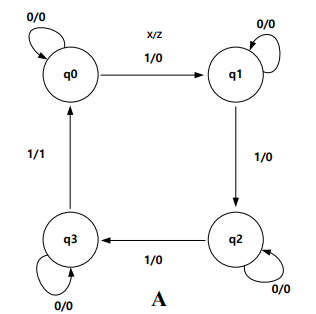
텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

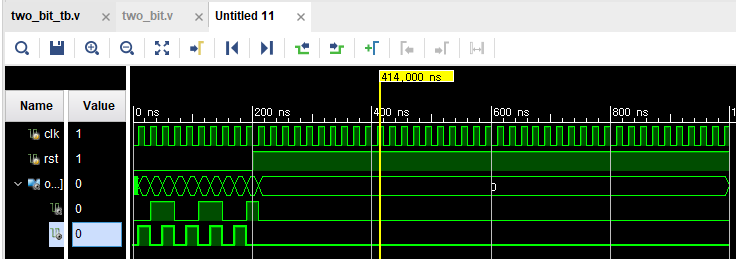
자동 생성된 설명



2bit 2진 카운터는 실험장님의 코드를 그대로 작성하였다. 이번에는 이전에 작성했던 코드들과는 조금 다르게 소스코드를 테스트 배치 파일처럼 작성하였다. 우선, 첫 번째로 기존에 module( ); 안에 input과 output을 선언하였지만 이번에는 괄호안에 사용할 변수명만 적은 후 괄호 밖에 input과 output을 정의하여 선언했다. 또한, assign을 하게되면 값이 계속해서 변하지 않기 때문에, output을 다시 reg로 선언해주었다. 이후 코드는 C언어에서의 작성법과 비슷했다. 우선, 변수 out의 초기값을 설정한다. 이번 실습에서는 초기 값이 00이므로 00으로 설정했다. 이후, always와 posedge를 활용해 반복문처럼 작성한다. clk가 상승에지가 될 때 마다 begin end 사이 구문을 실행한다. Rst는 reset을 나타내는 변수로 스위치가 켜지게 되면(high가 되면) 현재 out을 00으로 만든다. 그 외의 경우에는 +기호를 활용해 1비트씩 더해준다. 이전 실습들에서는 가산 역시 직접 코드로 작성하였지만, 이번 실습부터 +기호를 사용해도 된다고 한다. 이렇게 되면, verilog에서 자동으로 가산기 처럼 수행한다. 그래서 스케메틱 역시 위와 같이 그려지게 된다. 스캐메틱을 보면, 처음에 out값이 들어가고 이후 출력되는 out이 다시 입력으로 들어가는 모습을 하고 있다. 따라서 out은 00 -> 01 -> 10 -> 11 -> 00 …을 반복할 것이다. ( 2비트를 넘어갈 경우 무시하기 때문이다.) 따라서 주어진 상태도로부터 상태표는 아래 표와 같이 작성될 수있다.

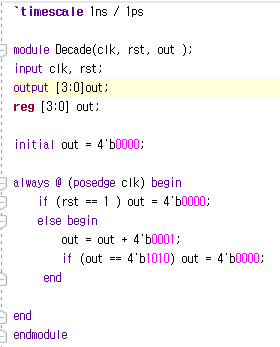


|  |  |  |
| --- | --- | --- |
| Present state qi | Next state/ output x/z | |
| Input x = 0 | Input x = 1 |
| q0 | q0 / 0 | q1 / 0 |
| q1 | q1 / 0 | q2 / 0 |
| q2 | q2 / 0 | q3 / 0 |
| q3 | q3 / 0 | q0 / 1 |

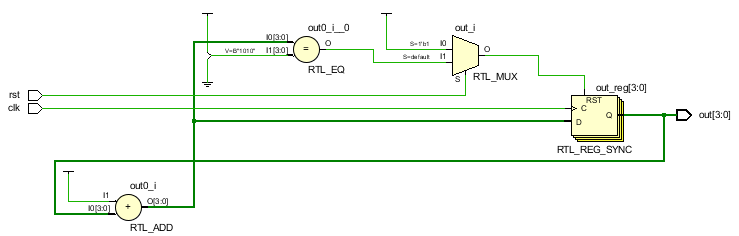


시뮬레이션을 확인해보면, 처음에 clk가 들어오기 전까지 out이 모두 00임을 볼 수 있다. 이후, 상승에지를 만나면, 01이되고, 그 다음 상승에지를 만나기 전까지 값을 유지한다. 다음 상승에지를 만나면 10이 되며 마찬가지로 값을 유지하다가 다시 11이 된다. 앞서 잠깐 말했듯이, +1을 계속 더해주어 원래는 100이 되야하지만, 2진 비트를 넘어가는 비트는 무시가 되므로 다시 00이 되어 반복되는 것을 볼 수 있다. 이제, rst가 1이되는 경우 00에서 값이 바뀌지 않는 것을 확인할 수 있다.

**2. 4-bit decade counter**

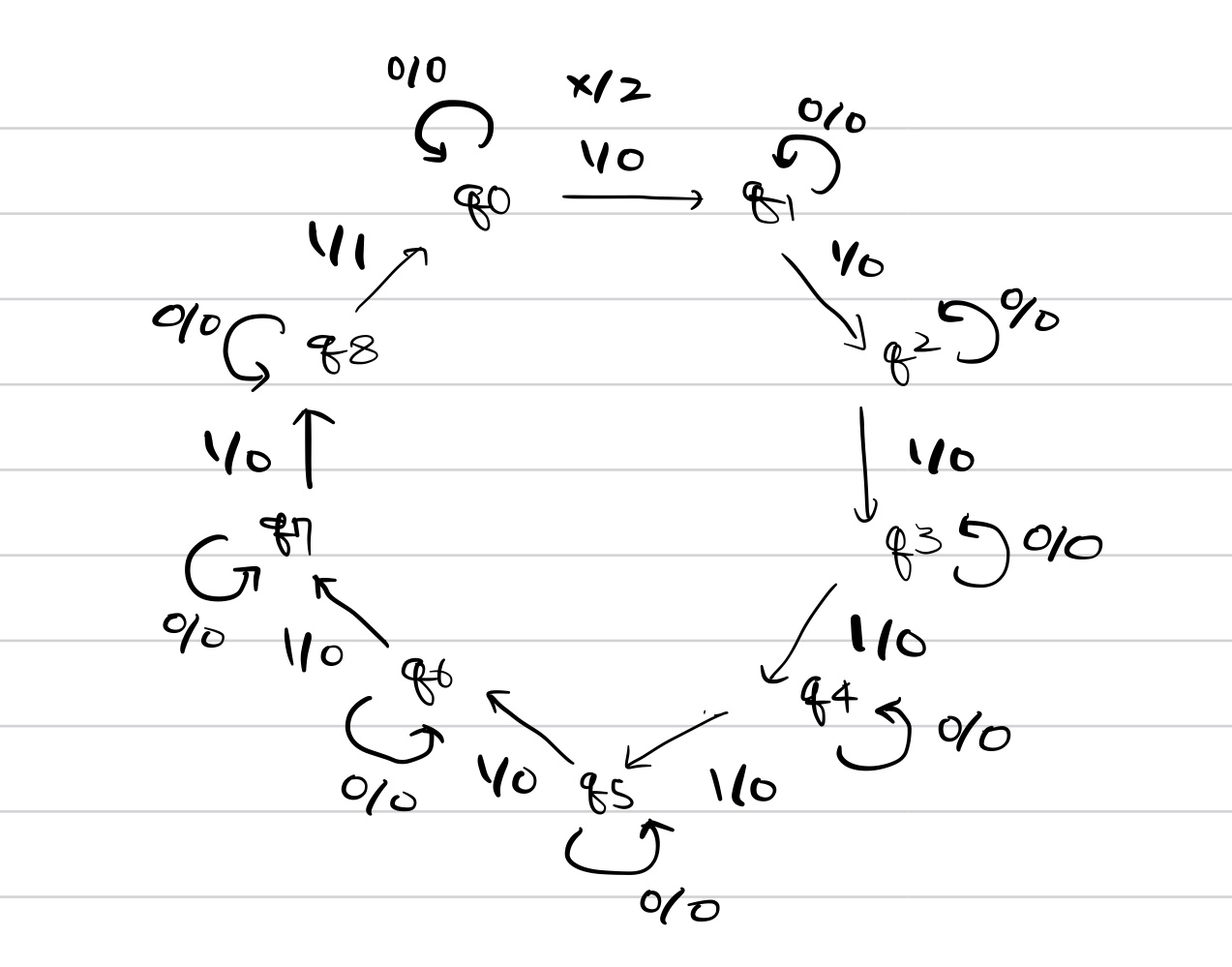
**텍스트이(가) 표시된 사진

자동 생성된 설명**

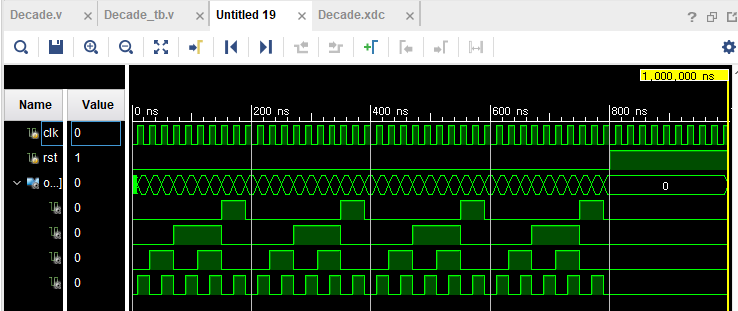
****

마찬가지로, 앞서 작성한 방법과 비슷하게 작성하였다. 이번 4-bit decade 카운터는 0~9까지 값이 순차적으로 올라간다. 다만, 4비트는 최대 15가 될 수 있으므로 예외처리가 필요하다. 우선, 출력에 사용할 변수 out을 배열로 선언하고 마찬가지로 reg로도 같이 선언해준다. 사용할 입력은 clk와 rst로 위와 동일하다. 초기 값은 0000이다. 이제, rst가 1이 되면 4개의 비트는 모두 0이 되어야 한다. 반면, rst가 0일 때는 1비트씩 더해주어 출력이 0000 -> 0001 -> 0010 -> 0011 -> 0100 -> 0101 -> 0110 -> 0111 -> 1000 -> 1001이 되도록 해준다. 이제, 1010이 되게 되면 출력을 모두 0으로 해준다. 이를 if else문을 적절히 활용해 코드를 작성했다. 따라서, 1001의 다음출력은 0000이다.

스케메틱을 보면, =으로 되어있는 부분에서 if문 out == 4’b1010부분을 확인한다. 이후MUX라고 되어 있는 부분에서 0000이 되는 조건들이 하나라도 충족되는지 확인하는 것 같다. (rst == 1 또는 if문의 true) 이제, 앞선 2비트 카운터와 마찬가지로 출력되는 out이 다시 입력 D로 들어가며 1을 더해준다. 따라서, 상태도 및 상태표는 아래와 같이 작성된다.



|  |  |  |
| --- | --- | --- |
| Present state qi | Next state / output | |
| Input x = 0 | Input x = 1 |
| q0 ( 0000 ) | q0 / 0 | q1 / 0 |
| q1 (0001) | q1 / 0 | q2 / 0 |
| q2 (0010) | q2 / 0 | q3 / 0 |
| q3 (0011) | q3 / 0 | q4 / 0 |
| q4 (0100) | q4 / 0 | q5 / 0 |
| q5 (0101) | q5 / 0 | q6 / 0 |
| q6 (0111) | q6 / 0 | q7 / 0 |
| q7 (1000) | q7 / 0 | q8 / 0 |
| q8 (1001) | q8 / 0 | q0 / 1 |

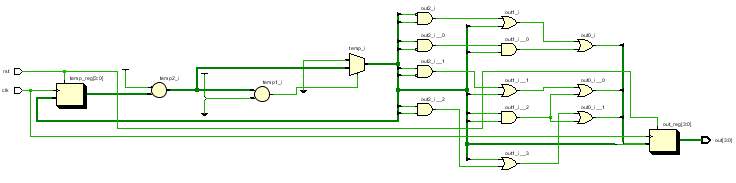
****

**3. 4-bit 2421 decade counter**

**텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명**

****

앞선 6주차에서 8421 BCD코드를 2421BCD 코드로 변환하는 실습을 진행한 적이 있다. 이와 비슷하게 8421 Decade 카운터를 2421 Decade 카운터로 만들어야 한다. 0~9까지를 출력하는 것은 동일하다. 아래 표는 8421코드와 2421코드의 진리표이다.

테이블이(가) 표시된 사진

자동 생성된 설명

가장 쉬운 방법은 5가 넘어갈 때 기존 비트에 0110을 더하는 것이다. 예를 들어 0101에 0110을 더하면 십진수 5값을 2421코드로 변환한 1011이 된다. 하지만, 이런 2421코드로 만드는 불 표현식을 직접 구하여 코드를 작성하는 것이 이번 실습의 목표이기 때문에 이런방식은 사용하지 않았다.

6주차에서 위의 진리표를 이용해 각 W,X,Y,Z를 카르노 맵을 사용하여 구했었으므로 이번 보고서에서 각 불 표현식을 구하는 과정은 생략하도록 한다.

W = A + BD + BC

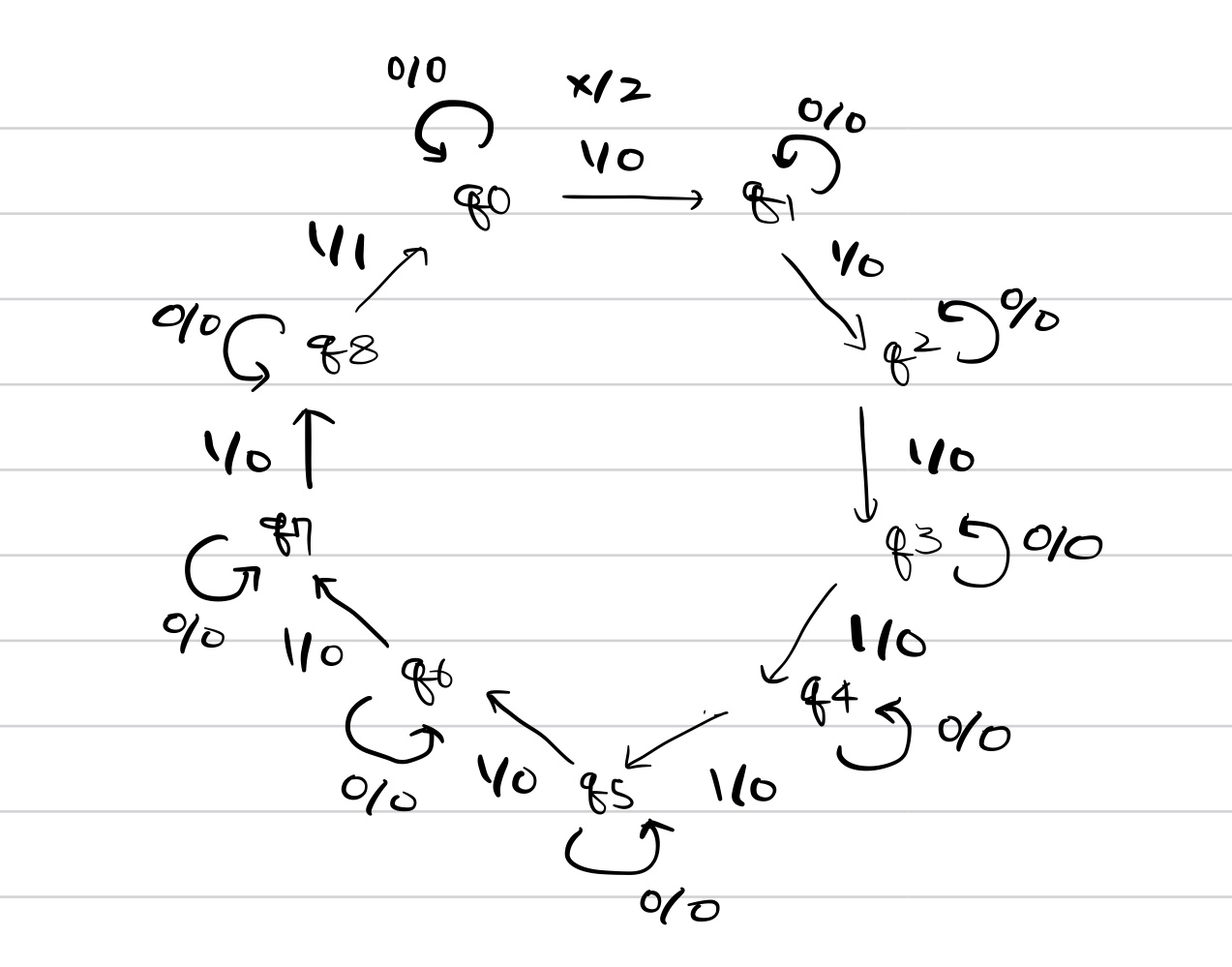
X = A + BC + BD’

Y = A + B’C + BC’D

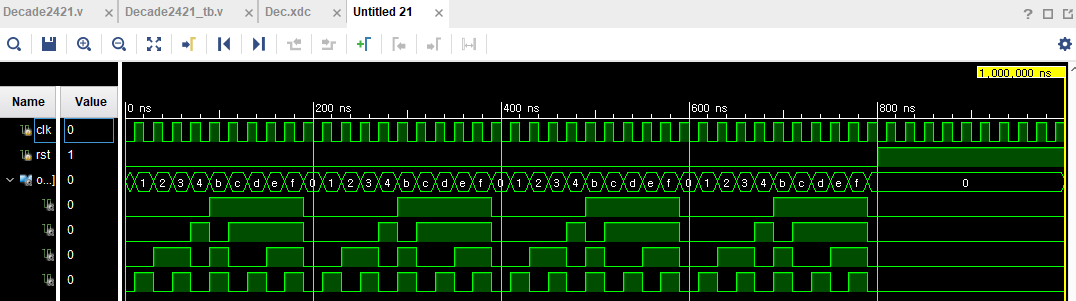
Z = D

위 코드에서 reg로 선언한 temp변수와 out변수를 사용했는데 temp변수는 C언어에서와 유사하게 임시변수를 나타낸다. 이렇게 작성한 이유는 위 표를 보면 알 수 있듯, 0100까지는 8421과 2421은 동일한 모습을 보여주기 때문에 예외 처리를 위해 사용했다.

이제 마찬가지로 초기 값으로 0000을 out과 temp모두에 할당한다. Temp 변수는 앞선 decade 카운터처럼 매 상승에지를 만날 때 마다 1비트씩 더한다. 또한, 마찬가지로 1010~1111이 나오면 안되므로 temp값이 1010이 되는 경우 temp의 모든 비트가 0이 된다. 이제, 위에서 구한 불 표현식을 사용한다. W를 out[3], X를 out[2], Y를 out[1] 그리고 Z를 out[0]으로 생각한다. A,B,C,D도 차례대로 temp[3],temp[2],temp[1],temp[0]으로 생각하면 위 코드와 같이 작성할 수 있다. Rst가 1이 되면, temp가 0000이되고 out도 함께 0000으로 해주어야한다.



|  |  |  |
| --- | --- | --- |
| Present state qi | Next state / output | |
| Input x = 0 | Input x = 1 |
| q0 (0000) | q0 / 0 | q1 / 0 |
| q1 (0001) | q1 / 0 | q2 / 0 |
| q2 (0010) | q2 / 0 | q3 / 0 |
| q3 (0011) | q3 / 0 | q4 / 0 |
| q4 (0100) | q4 / 0 | q5 / 0 |
| q5 (1011) | q5 / 0 | q6 / 0 |
| q6 (1100) | q6 / 0 | q7 / 0 |
| q7 (1101) | q7 / 0 | q8 / 0 |
| q8 (1111) | q8 / 0 | q0 / 1 |

****

시뮬레이션 결과를 확인해보면, 의도한 대로 잘 나온 것을 볼 수 있다. 우선, 처음엔 초기값 0000이다. 이후, 상승에지를 만나면 0001을 다음 상승에지를 만나기전까지 유지한다. 다음은 0010, 0011, 0100이다. 이후 기존 decade 카운터는 0101이지만, 2421 코드로 바꾸면 1011이며 시뮬레이션상에서 역시 1011이 나오는 것을 볼 수 있다. 다음 역시 1100, 1101, 1110, 1111 순으로 출력결과가 나오는 것을 확인할 수 있다. 이제 0~9까지 모두 순차적으로 등장했으므로 다시 0000이 나오는 것 역시 확인할 수 있으며 rst가 1이되기 전까지 이를 반복하는 것을 볼 수 있다. Rst가 1이 되면 모든 값이 0000으로 설정되며 이를 계속 유지하는 것을 볼 수 있다.

**4. 결과 검토 및 논의**

이번 실습에서는 새로운 방식으로 코드를 작성하였다. 이전과는 다르게, if else문을 사용하였고 always문과 posedge를 활용하였다. 하지만, 처음엔 익숙하지 않아 2421 decade 카운터에서 잦은 실수를 반복하였다. 임시 변수 temp를 다시 out에 할당하여 하는 방식을 사용했는데 이 과정에서 rst가 들어왔을 때 temp만 0으로 초기화 한다던지 등의 이유로 실습이 원할하지 않았었다.

또한, FPGA보드에서 정상적으로 작동시키기 위해서 클락 변수 clk를 xdc파일에 다음 문구를 추가해야했다.

set\_property ALLOW\_COMBINATORIAL\_LOOPS true [get\_nets <net\_name>]

set\_property SEVERITY {Warning} [get\_drc\_checks LUTLP-1]

set\_property SEVERITY {Warning} [get\_drc\_checks NSTD-1]

set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets clk\_IBUF]

이전 11주차 실습에서 구현했던 플리플롭의 경우 always문을 사용하지 않아 완벽하게 원했던 시뮬레이션 결과가 나오지는 않았지만, 이번 실습의 경우에는 원하는 대로 구현을 할 수 있었다.

clk의 상승에지를 만날 때 마다 2진 카운터의 경우 00 -> 01 -> 10 -> 11 -> 00 … 순으로 반복하였고, decade 카운터는 0~9까지, 그리고 9이후에는 다시 0으로 바뀌어 반복하였다. 2421 카운터 decade 카운터와 유사하게 출력 역시 2421 코드의 0~9를 반복하였으며, decade 카운터와 상태도는 거의 동일하게 나타났다. 시뮬레이션 결과를 확인할 때 배열로 선언하여 확인하기가 더 쉬웠으며 FPGA보드로의 확인도 쉬웠다. 입력은 clk와 rst밖에 없기도 하였고 4개의 스위치가 0부터 1씩 증가하여 차례대로 작동했기 때문이다. 또한, 매 clk 마다 모든 비트가 변화하므로 동기식으로 볼 수 있을 것 같다.

하지만, counter는 보통 플리플롭으로 구성되는 것으로 알고있다. 이번 실습에서는 단순하게 코드를 통해 counter를 구현한점이 조금 아쉽다고 생각한다. 디지털회로개론에서는 상태도를 이용해 상태표를 얻은 다음 사용할 플리플롭을 결정하고 각 상태의 binary 값을 지정해준다. 이후, 여기표라 불리는 excitation table을 이용해 출력함수를 얻어낼 수 있었다. 이를 바탕으로 최종적으로 회로를 구성할 수 있다.

**5. 추가 이론 조사 및 작성**

위에서 잠깐 언급했듯이 플리플롭을 활용해 2진 카운터나 BCD up/down 카운터 등을 구현할 수 있다. T 플리플롭을 활용해 4bit down 비동기식 카운터를 구현해 보면 다음과 같다. reset은 구현하지 않았다.

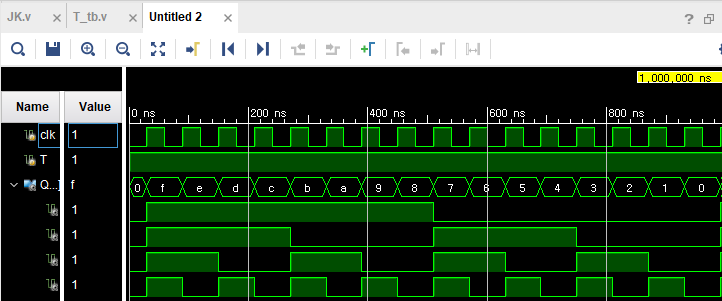
텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

보통은 4비트 이므로 현재 상태 A,B,C,D에 대해 다음 상태 A+ B+ C+ D+의 진리표를 작성한다. 이후, 4개의 T플리플롭 Ta Tb Tc Td의 여기표를 작성하고 카르노맵을 이용하면 회로를 만들 수 있다. 하지만, 비동기식 카운터를 T플리플롭으로 만들 경우 조금만 생각하면 위 코드와 같이 될 수 있다는 것을 알 수 있다.

우선 이 감소 카운터의 경우 처음에 0000에서 상승에지를 만나면 Q[0]이 1이되고 Q[0]이 1이되면 Q[1], 그리고 Q[2], Q[3]가 차례로 1이 된다. 1111에서 1110이 되려면, Q[0]이 다음 상승에지를 만나 토글(1에서 0)이 되어야한다. 그러나, 나머지 Q[3], Q[2], Q[1]은 변하지 않는다. 이런식으로 쭉 가다보면, 자연스럽게 1111에서 0000으로 바뀌는 것을 알 수 있으며 이를 반복하는 것을 유추할 수 있다.



마찬가지로, 시뮬레이션을 확인해보면 f부터 순차적으로 상승에지를 만날 때 마다 0까지 내려가는 것을 볼 수 있다. 또한, 코드에서 보면, clk가 되는 부분이 Q[0], Q[1], Q[2]가 되어 비동기식임을 알 수 있다.

