**12주차 결과보고서**

**전공: 생명과학과 학년: 4학년 학번: 20182186 이름: 김승원**

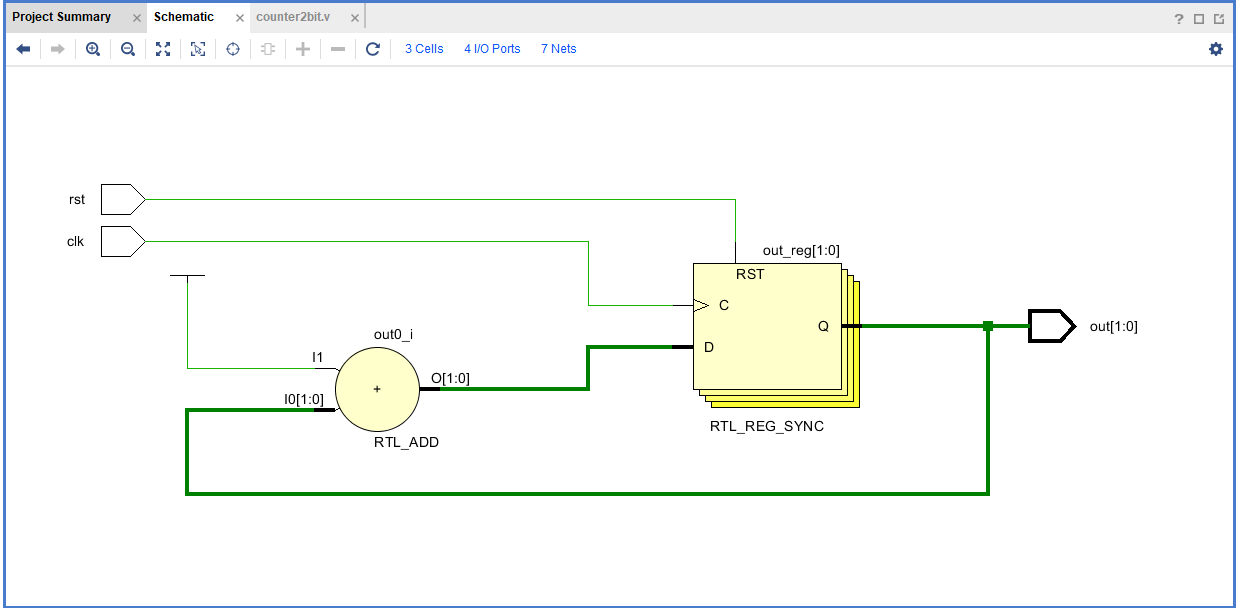
**1. 2-bit counter의 결과 및 simulation 과정 설명(Verilog source code, 출력 예시/schematic 포함, 과정 상세히)**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**<Figure 1. 2-bit counter의 source code>**

2-bit counter는 00, 01, 10, 11을 반복하는 counter이다. 우선 clock과 reset을 input으로 입력하고 out을 배열을 사용하여 output으로 입력했다. Reg로 out을 선언해주고, initial문을 사용해서 out을 00에서 시작하도록 강제로 값지정을 하였다. 또한 rising edge를 사용하기 위해 always posedge clk를 사용했고, rst input(RESET)이 1인 경우 output을 00으로 고정시키고, 그렇지 않은 경우 1bit씩 증가시켜 counter를 구현했다.

****

**<Figure 2. Schematic 결과>**

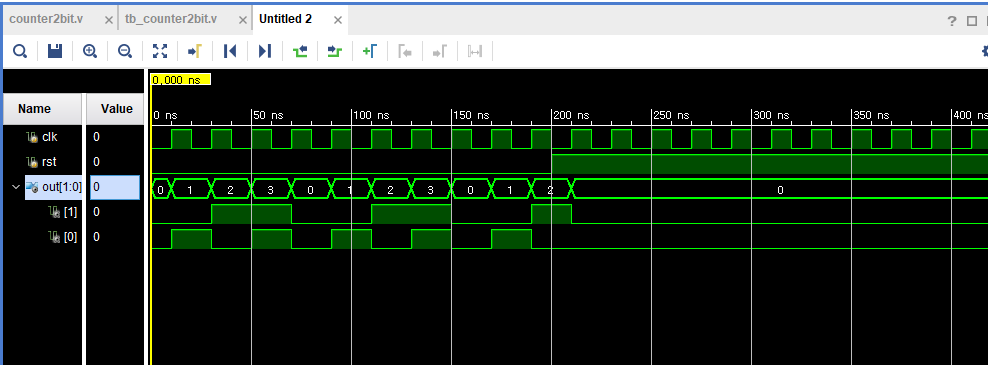
Schematic 결과로 D flip flop이 사용된 counter의 모습을 확인했다. Reset인 rst가 flipflop에 잘 연결되어 있고, clock input인 clk도 잘 연결되어 있었고, 출력도 옳게 잘 구현된 것을 확인했다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**<Figure 3. 2-bit counter test bench code>**

위의 code를 통하여 simulation 결과를 다음과 같이 얻었다.

****

**<Figure 4. 2-bit counter simulation 결과>**

2-bit counter의 simulation결과이다. 우선 rst신호가 0일 때 out 결과값이 잘 나온 것을 확인했고, 앞의 source code에서 posedge clk를 활용해 구현하였으므로, rising edge에서 out 값이 업데이트 된 것을 확인했다. 우선 시작값 00으로 고정해서 시작했으므로, 맨 먼저 00이 나왔고, 그 다음 01, 10, 11 순으로 확인됐다. 그리고 00으로 돌아가 다시 01, 10, 11을 반복하는 것으로 보아 counter가 잘 구현되었음을 simulation 결과를 통해 확인했다. FPGA 보드 결과 역시 clk switch를 키고 끄고 반복했을 때 00, 01, 10, 11, 00, 01, 10, 11 순서로 불이 들어오는 것을 확인했다. 그리고 rst switch를 킨 다음 clk switch를 올릴 경우 다시 00으로 reset되는 것을 확인하였다.

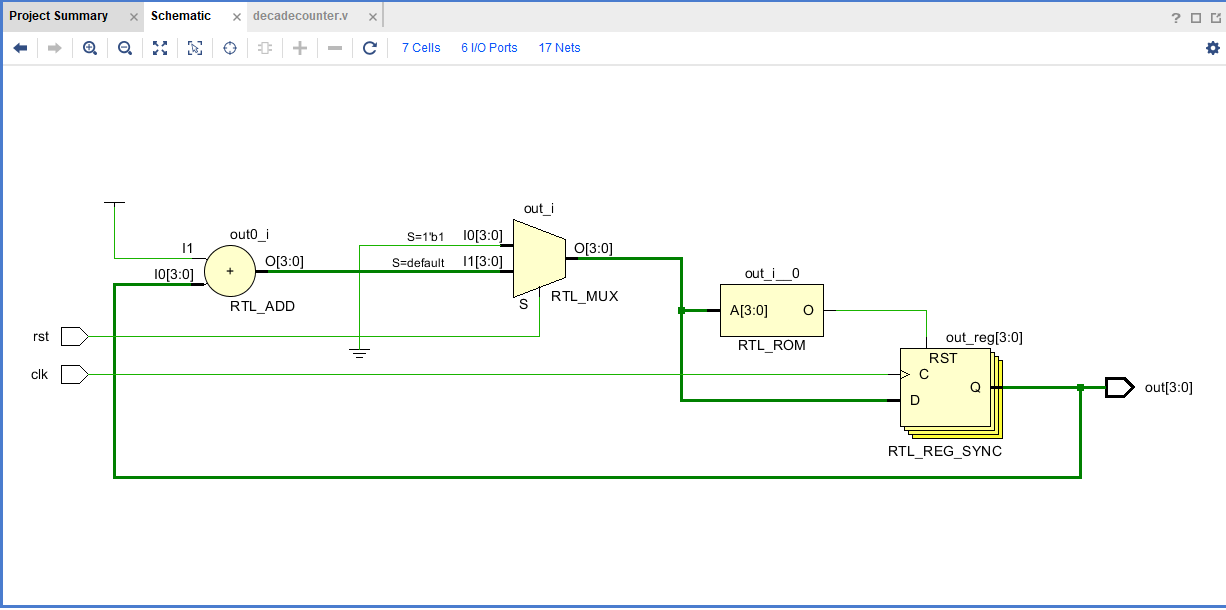
**2. 4-bit decade counter의 결과 및 simulation 과정 설명(Verilog source code, 출력 예시/schematic 포함, 과정 상세히)**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**<Figure 5. Decade counter의 source code>**

Decade counter를 설계하기 위해서 clk, rst(reset)을 input으로 입력하고, output 출력을 out을 설정했다. Out은 배열을 사용하여 4bit를 구현하고자 했고, out의 초기값은 0000으로 설정해주었다. 그리고 Rising edge에서 값을 업데이트 하기 위해 posedge clk를 사용했다. If 문을 활용해서 rst가 1이면 0000으로 초기화시켰다. 그리고 rst가 1이 아닌 경우 out이씩 증가하도록 했다. 그리고 if문을 활용해서 out값이 1010(10)이 되면 out의 값을 0000으로 돌아가도록 했다.

****

**<Figure 6. Decade counter schematic 결과>**

Decade counter schematic 결과이다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**<Figure 7. decade counter test bench code>**

위의 code를 통하여 simulation 결과를 다음과 같이 얻었다.

**텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명**

**<Figure 8. decade counter simulation 결과>**

Simulation 결과를 확인해보면, 우선 초기값은 0000으로 설정한대로 나왔다. 또한 decade counter는 1씩 증가하다가 1001이 넘어가면 0000으로 초기화 시켜주어야 하므로, 약 190ns 시점에서 0000으로 잘 돌아온 것이 확인됐다. 또한 source code에서 rising edge에서 값이 업데이트 되도록 코딩했으므로, 출력 값이 rising edge에서만 업데이트 됐다. 결과값을 확인한 후 rst(RESET) 값을 1로 고정시킨 시점부터는 출력 값이 0000으로 아무것도 나오지 않았다.

**3. 4-bit 2421 decade counter의 결과 및 simulation 과정 설명(Verilog source code, 출력 예시/schematic 포함, 과정 상세히)**

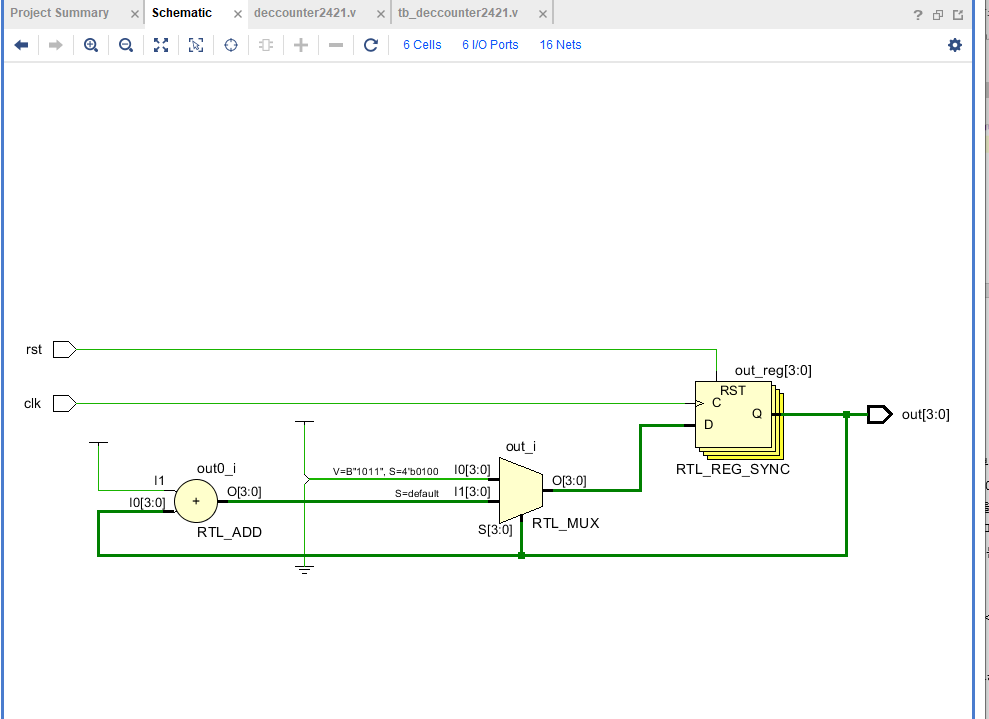
**텍스트이(가) 표시된 사진

자동 생성된 설명**

**<Figure 9. 2421 decade counter의 source code>**

2421 decade counter는 decade counter와 비슷하다. 다만 decimal number 0부터 4까지는 동일한 2진수로 표현되지만, 5부터 9까지 다르게 표현된다. 2421 BCD의 경우 5는 1011, 6은 1100, 7은 1101, 8은 1110, 9는 1111이다. 따라서 tmp 변수를 따로 설정해준 다음 count가 4에 해당하는 2진수에 도착했을 때 8421 BCD에서 2421로 위와 같은 식을 통해 전환해주었다. 앞서 보았던 BCD counter와 나머지 부분은 동일하다. 1씩 카운트되면서 올라가고, 4에 해당하는 2진수가 나오면 2421로 변환해주는 방법을 사용했다.

그리고 clk 신호가 rising edge일 때 count가 되도록 설계하였고, rst가 1이면 출력값이 0, rst가 0일 때 output의 변화가 생기도록 설계했다.

****

**<Figure 10. 2421 Decade counter schematic 결과>**

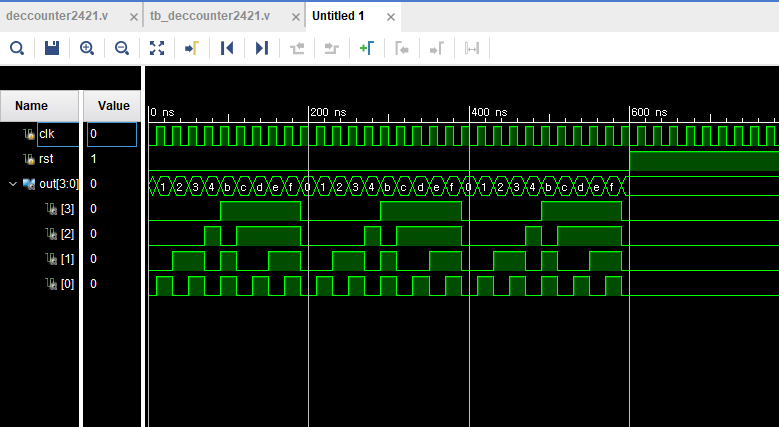
2421 decade counter schematic의 결과이다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**<Figure 11. 2421 decade counter test bench code>**

위의 code를 통하여 simulation 결과를 다음과 같이 얻었다.

****

**<Figure 11. 2421 decade counter simulation 결과>**

우선 source code에서 확인할 수 있듯이, rising edge에서 update되도록 설계하였으므로, 위와 같이 rising edge에서 count가 진행됐다. 먼저 초기값은 0000이고, 이후 0001, 0010, 0011, 0100, 1011, 1100, 1101, 1110, 1111로 2421 BCD counter로 잘 설계되었음을 확인했다.

**4. 결과 검토 및 논의 사항**

본 실험에서 초기값을 설정할 때 직접적으로 out = 4’b0000과 같이 사용해서 초기값을 0으로 고정시키고 시작했는데, 임의의 값에서 count가 시작되지 않도록 rst 입력을 사용해서도 0으로 고정시키는 것이 가능하다고 생각된다. 또한 clk 신호의 경우 rising edge일 때 마다 always문을 실행하도록 설계했는데, posedge !clk와 같이 사용해서 falling edge에서 always 문이 항상 실행되도록 할 수 있다.

또한 일반적인 n-bit binary counter의 경우 단순히 0000부터 0001, … , 1110, 1111까지 반복된다. 여기서 rst가 1이 아니면 항상 output이 1씩 증가하도록 설계하면 되는데, BCD counter와 2421 BCD counter는 현재 출력 값이 9이거나 4일 경우 0000이나 1011으로 바꿔주면서 중간 값들을 건너 뛰도록 해야 하는 과정이 필요하다. 즉, if 문이나 if else문을 활용해서 현재 상태에 따라 다음 출력 값이 어떻게 출력되도록 할 것 인지 결정해주도록 해야 한다.

Simulation 결과와 FPGA board의 동작을 확인했을 때 모두 결과를 올바르게 얻었기 때문에 source code를 정확하게 설계했다고 판단된다.

**5. 추가 이론 조사 및 작성**

Divide-by-N counter는 N가지 상태를 순서대로 반복하는 카운터다. 2진 카운트 순을 따르기도 하지만 다른 임의의 순서를 따르기도 한다. 디지털 시스템에서 counter는 동작 순서를 제어하기 위한 timing signals를 생성하는데 사용된다. Counter는 시프트 레지스터를 이용해 구현되기도 한다. 2진 counter 외에 ring counter과 johnson counter 등이 있다. Ring counter는 임의의 시간에 하나의 flip-flop만 1이 되고, 나머지의 모든 flip-flop은 모두 clear가 되는 circular shift Register이다. N- bit ring counter는 flip-flop들 사이에 단일 비트를 순환시켜 n개의 다른 상태를 만든다. 만약 shift register이 switch-tail ring counter로서 연결되면 상태의 수를 2배로 늘릴 수 있는데, johnson counter는 2n개의 timing signals 출력을 만들기 위하여 2n개의 decoding gate를 가진 n-bit switch-tail ring counter라 할 수 있다.