2018/10/07

Counter & shiftReg 과제 보고서

디지털 시스템 설계

21700034 곽영혜

* 목차

서론

과제의 목적

본론

1. 설계한 모듈의 기능
2. 코드 설명

Counter1.v (비동기 reset>inc)

Counter2.v (비동기 reset>inc>load)

Counter3.v (동기식 reset>load>inc)

shiftReg1.v (load>shiftR)

shiftReg2.v (비동기 reset>load>shiftR)

shiftReg3.v (동기식 reset>shiftR>shiftL)

Testbench.v (test of 6 modules)

1. Testbench waveform 설명

결론

설계 결과 및 요약

* 과제의 목적

Counter

: 비동기식 카운터와 동기식의 카운터의 차이와 제어 신호의 우선순위에 따라 다르게 나타나는 결과값(Q[7:0])을 확인하여 카운터의 작동 원리를 파악하는 것입니다.

: 카운터는 8-bit의 입력값과 출력값을 가지게 되며, 제어 신호가 2~3개를 가지게 됩니다. 이 제어 신호의 우선순위와 동기식 카운터, 비동기식 카운터의 경우를 다르게 하여 총 3개의 카운터 모듈(counter1, 2, 3)를 가지게 됩니다.

shiftReg

: 비동기식 shiftRegister과 동기식의 shiftRegister의 차이와 제어 신호의 우선순위에 따라 다르게 나타나는 결과값(Sout=Q[0])을 확인하여 shiftRegister의 작동 원리를 파악하는 것입니다.

: shiftRegister는 8-bit의 입력값과 1-bit의 출력값을 가지게 되며, 제어 신호가 2~3개를 가지게 됩니다. 이 제어 신호의 우선순위와 동기식 shiftRegister, 비동기식 shiftRegister의 경우를 다르게 하여 총 3개의 shiftRegister 모듈(shiftRegister1, 2, 3)를 가지게 됩니다.

: 특히 shiftRegister1, 2는 오른쪽에서 옮겨지는 경우만 있지만, shiftRegister3은 왼쪽에서 옯겨지는 경우도 있습니다.

1. 설계한 모듈의 기능

Counter1 (비동기 reset>inc)

: 입력값과 상관없이 reset신호와 increase신호를 통해 counter값이 리셋되거나 증가하는 카운터입니다.

: 만약 reset신호가 0이 되면 카운터의 값이 리셋되도록 설정하고 다른 경우들 중 만약 increase신호가 1이 되면 카운터의 값이 증가하도록 설정합니다.

: 단, 이 모든 경우가 clk이 증가할 때(posedge) 일어나야 됩니다.

Counter2 (비동기 reset>inc>load)

: reset신호와 increase신호, load신호를 통해 counter값이 리셋되거나 증가, 입력값이 들어가는 카운터입니다.

: reset과 increase신호의 원리는 위의 counter1과 동일하게 하며, load신호가 1이 되면 입력값이 counter값으로 들어가도록 설정합니다.

: 단, 이 모든 경우가 clk이 증가할 때(posedge) 일어나야 됩니다.

Counter3 (동기식 reset>load>inc)

: reset신호와 increase신호, load신호를 통해 counter값이 리셋되거나 증가, 입력값이 들어가는 카운터입니다.

: 위의 counter2의 원리와 동일하지만 increase신호와 load신호의 우선순위가 다릅니다.

: 단, 이 모든 경우가 clk이 증가할 때(posedge)나 reset이 감소할 때(negedge) 일어나야 됩니다.

shiftReg1 (load>shiftR)

: load신호와 shiftR신호를 통해 Q값에 입력값이 들어가거나 오른쪽으로 옮겨져 출력값(Sout=Q[0])이 결정되는 구조입니다.

: 만약 load신호가 1이 되면 입력값이 들어가도록 설정하고 다른 경우들 중 만약 shiftR신호가 1이 되면 오른쪽에서 1칸씩 값이 이동하도록 설정합니다.

: 단, 이 모든 경우가 clk이 증가할 때(posedge) 일어나야 됩니다.

shiftReg2 (비동기 reset>load>shiftR)

: reset신호와 load신호, shiftR신호를 통해 입력값이 리셋되거나 들어가거나 오른쪽으로 옮겨져 출력값(Sout=Q[0])이 결정되는 구조입니다.

: reset신호가 0이 되면 Q값이 0이 되도록 설정하며 나머지(load, shiftR신호)는 위의 과정과 동일합니다.

: 단, 이 모든 경우가 clk이 증가할 때(posedge) 일어나야 됩니다.

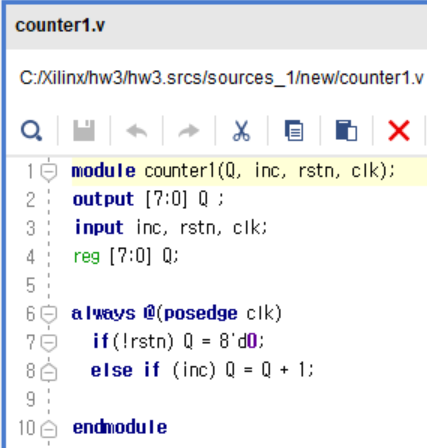
shiftReg3 (동기식 reset>shiftR>shiftL)

: reset신호와 load신호, shiftR신호를 통해 입력값이 리셋되거나 오른쪽, 혹은 왼쪽으로 옮겨져 출력값(Sout=Q[0])이 결정되는 구조입니다.

: reset과 shiftR신호의 경우, 위의 과정과 동일하지만 load신호 대신 shiftL신호가 오게 되었으며 1이 되면 왼쪽에서 1칸씩 값이 이동하도록 설정합니다. 또한 우선순위가 맨 뒤가 되도록 설정합니다.

: 단, 이 모든 경우가 clk이 증가할 때(posedge)나 reset이 감소할 때(negedge) 일어나야 됩니다.

1. 코드 설명
   * + Counter1.v



output[7:0] Q : 출력되는 결과값을 저장하는 변수

Input inc : counter값의 증가를 결정하는 변수

Input rstn : reset값을 받아오는 변수

Input clk : 클록 주기값을 받아오는 변수

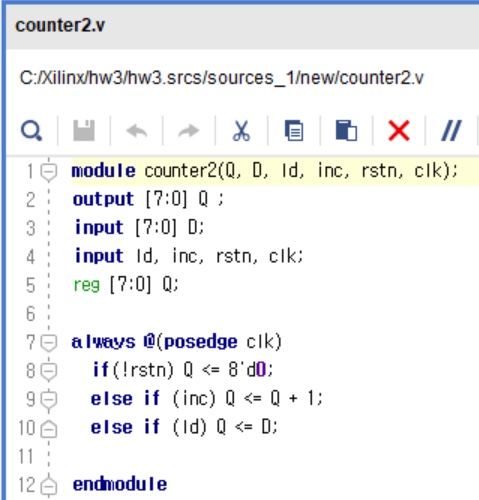
Reg[7:0] Q : 값이 증가하는 경우, 바뀐 값이 저장될 수 있도록 reg로 설정합니다.

Clk주기가 증가할 때, 출력값이 바뀔 수 있도록 always로 설정하며

If과 else if문으로 제어 신호의 우선 순위를 나타냅니다.

Rstn(reset)의 값이 0이 되는 경우에는 출력값이 초기화(0)되어야 하므로 Q에 0을 대입하며, inc의 값이 1이 되는 경우는 숫자가 카운트되었다는 의미이므로 +1을 하여 출력값을 증가시켜줍니다.

* + - Counter2.v



output[7:0] Q / Input inc / Input rstn / Input clk / Reg[7:0] Q

: 위(counter1)와 동일한 의미를 가집니다.

+ input[7:0] D : 입력되는 입력값을 받아와 저장하는 변수

+ input ld : 입력값을 불러와 출력값에 저장할 지 결정하는 변수

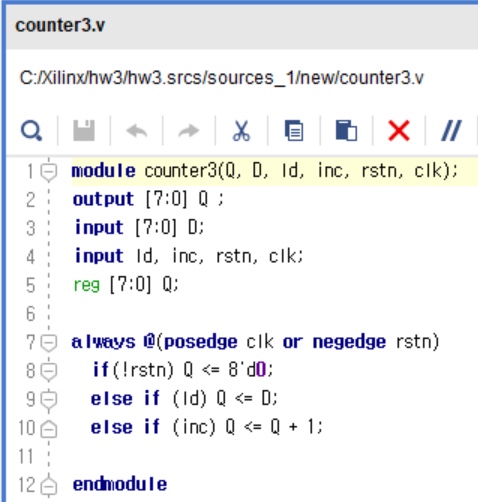
Clk주기가 증가할 때, 출력값이 바뀔 수 있도록 always로 설정하며

If과 else if문으로 제어 신호의 우선 순위를 나타냅니다.

Rstn(reset)의 값이 0이 되는 경우에는 출력값이 초기화(0)되어야 하므로 Q에 0을 대입하며, inc의 값이 1이 되는 경우는 숫자가 카운트되었다는 의미이므로 +1을 하여 출력값을 증가시켜줍니다.

위의 코드(counter1)와 달리 ld신호가 1이 되는 경우, 출력값에 기존의 입력값을 대입하여 출력하도록 설정합니다.

* + - Counter3.v



output[7:0] Q / Input[7:0] D / Input ld / Input inc / Input rstn / Input clk / Reg[7:0] Q

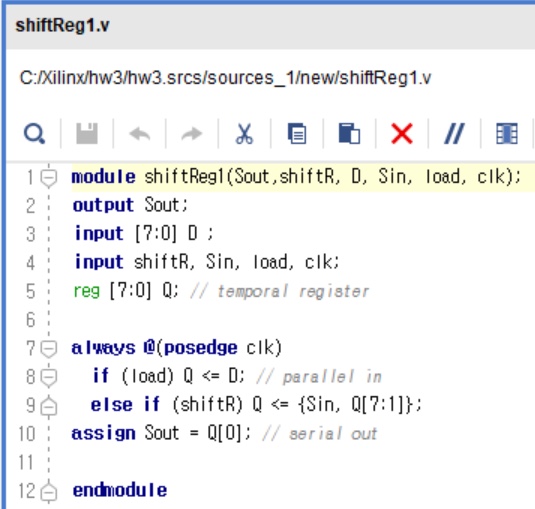
: 위(counter2)와 동일한 의미를 가집니다.

Clk주기가 증가할 때(poeedge)나 reset값이 감소할 때(negedge), 출력값이 바뀔 수 있도록 always로 설정하며 If과 else if문으로 제어 신호의 우선 순위를 나타냅니다.

Rstn(reset)의 값이 0이 되는 경우에는 출력값이 초기화(0)되어야 하므로 Q에 0을 대입하며, inc의 값이 1이 되는 경우는 숫자가 카운트되었다는 의미이므로 +1을 하여 출력값을 증가시켜주고 ld신호가 1이 되는 경우, 출력값에 기존의 입력값을 대입하여 출력하도록 설정합니다.

위의 코드(counter2)와 달리 동기식 reset을 제어 신호로 가지며 ld신호와 inc신호의 우선순위가 바뀌어 있어 출력 결과가 다르게 나타납니다.

* + - shiftReg1.v



Output Sout : 임시로 저장된 결과값 중 첫번째 bit의 값을 출력하는 변수

Input[7:0] D : 입력되는 입력값을 받아와 저장하는 변수

Input shiftR : 숫자가 오른쪽에서 1칸씩 밀려나는 것을 결정하는 변수

Input Sin : 숫자가 밀려날 경우, 빈 칸(빈 bit자리)를 채우는 변수

Input load : 입력값을 불러와 출력값에 저장할 지 결정하는 변수

Input clk : 클록 주기값을 받아오는 변수

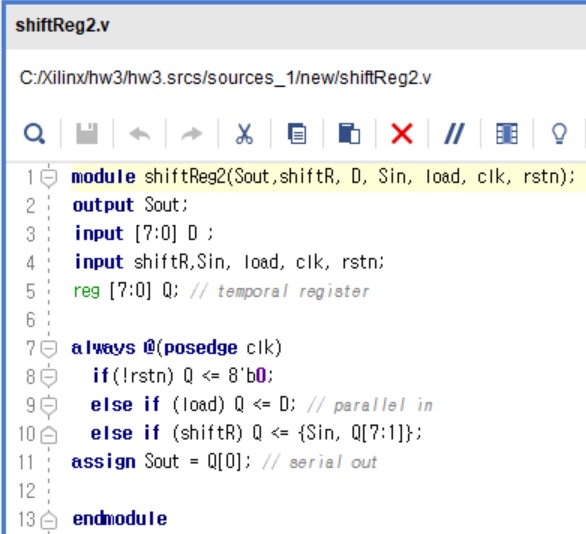
Reg[7:0] Q : 숫자가 밀려나는 경우, 바뀐 값이 임시로 저장될 수 있도록 reg[7:0]으로 설정합니다.

Clk주기가 증가할 때(poeedge) 출력값이 바뀔 수 있도록 always로 설정하며 If과 else if문으로 제어 신호의 우선 순위를 나타냅니다.

만약 Load신호가 1일 경우 입력값을 출력할 수 있도록 저장하며, shiftR신호가 1일 경우 Q값이 오른쪽에서 1칸씩 이동하고 빈 공간인 Q[0]에 Sin이 들어오도록 설정합니다.

마지막에 assign으로 출력값을 저장하여 Sout이 출력되도록 합니다.

* + - shiftReg2.v



Output Sout / Input[7:0] D / Input shiftR / Input Sin / Input load / Input clk / Reg[7:0] Q

: 위(shiftReg1)와 동일한 의미를 가집니다.

+ input rstn : reset값을 받아오는 변수

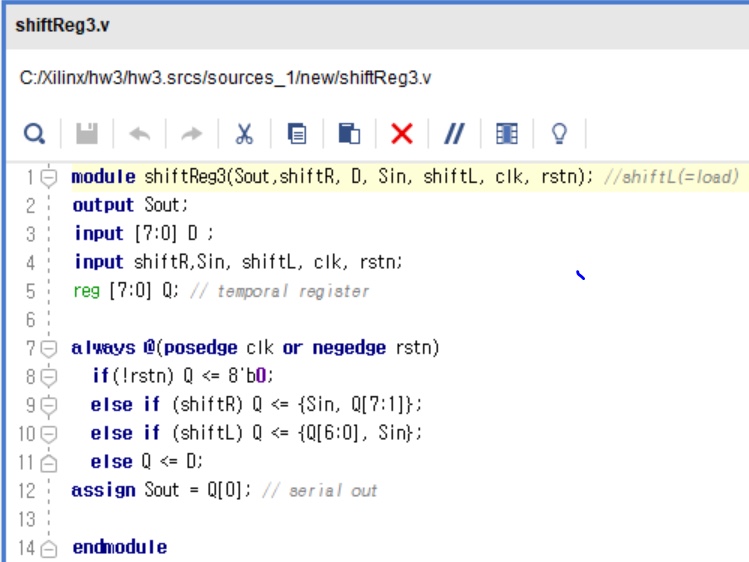
Clk주기가 증가할 때(poeedge) 출력값이 바뀔 수 있도록 always로 설정하며 If과 else if문으로 제어 신호의 우선 순위를 나타냅니다.

Load신호와 shiftR신호는 위의 코드(shiftReg1)와 동일하지만 비동기rstn신호가 우선순위 상단에 새롭게 추가되었습니다.

만약 rstn(reset)의 값이 0이 되는 경우에는 출력값이 초기화(0)되어야 하므로 Q에 0을 대입하도록 설정합니다.

마지막에 assign으로 출력값을 저장하여 Sout이 출력되도록 합니다.

* + - shiftReg3.v



Output Sout / Input[7:0] D / Input shiftR / Input Sin / Input clk / Reg[7:0] Q

: 위(shiftReg2)와 동일한 의미를 가집니다.

+ input shiftL : 숫자가 왼쪽에서 1칸씩 밀려나는 것을 결정하는 변수

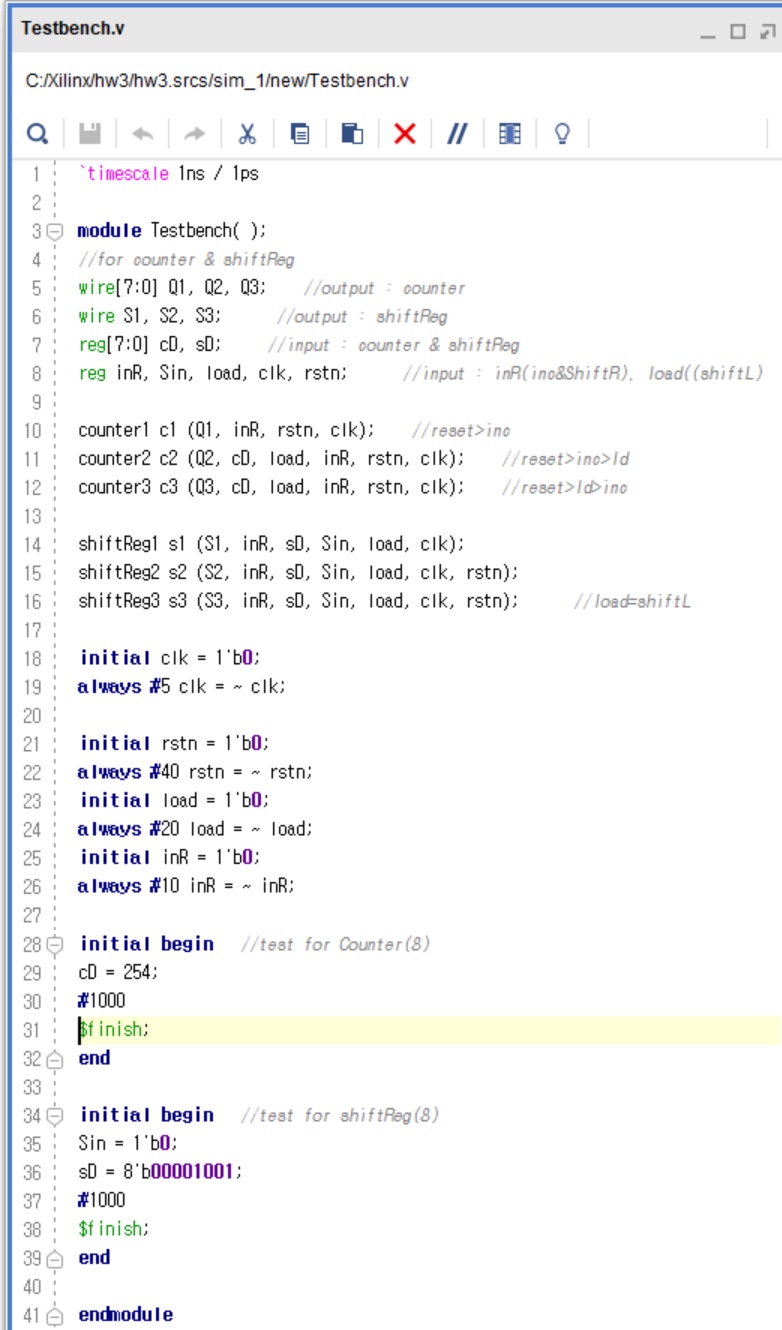
Clk주기가 증가할 때(poeedge)나 reset값이 감소할 때(negedge), 출력값이 바뀔 수 있도록 always로 설정하며 If과 else if문으로 제어 신호의 우선 순위를 나타냅니다.

rstn신호와 shiftR신호는 위의 코드(shiftReg2)와 동일하지만 load신호가 아닌 shiftL신호가 새롭게 추가되었습니다.

shiftL신호가 1일 경우 Q값이 왼쪽에서 1칸씩 이동하고 빈 공간인 Q[7]에 Sin이 들어오도록 설정합니다.

위의 코드(shiftReg2)와 달리 동기식 reset을 제어 신호로 가지며 shiftL이 추가됨에 따라 좌로 1칸씩 밀려날 수 있게 됩니다.

* + - Testbench.v



Wire[7:0] Q1, Q2, Q3 : counter1, 2, 3의 출력값들을 저장하는 변수

Wire S1, S2, S3 : shiftReg1, 2, 3의 출력값들을 저장하는 변수

Reg[7:0] cD, sD : counter와 shiftReg의 입력값들을 저장하는 변수

Reg inR : counter의 inc, shiftReg의 shiftR 신호의 값을 받는 변수

Reg Sin : shiftReg의 Sin(빈 bit 자리 채우기) 신호의 값을 받는 변수

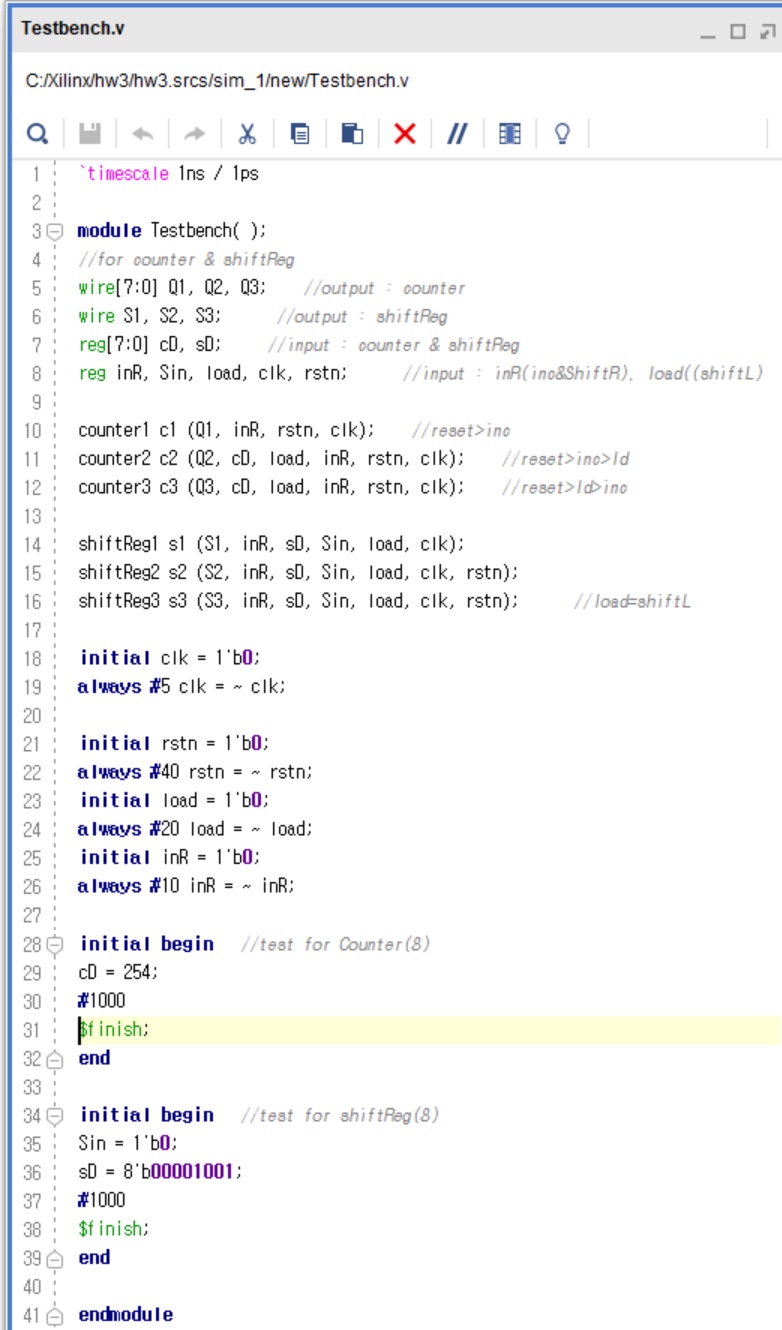
Reg load : counter와 shiftReg의 load 신호의 값을 받는 변수

Reg clk : counter와 shiftReg의 clk(클록) 신호의 값을 받는 변수

Reg rstn : counter와 shiftReg의 reset 신호의 값을 받는 변수

Module 6개 선언(c1, c2, c3 / s1, s2, s3)

초기 Clk(클록)값을 0으로 하며 주기를 5ns로 반복적으로 나타나도록 설정합니다.



초기 rstn과 load, inR값을 0으로 설정하고 inR은 10ns, load는 20ns, rstn은 40ns 단위로 설정하여 8가지의 경우의 수를 만듭니다.

<진리표>

|  |  |  |
| --- | --- | --- |
| rstn | load | inR |
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |

Initial begin을 2개 선언하여

cD = 254는 counter1, 2, 3을 위해

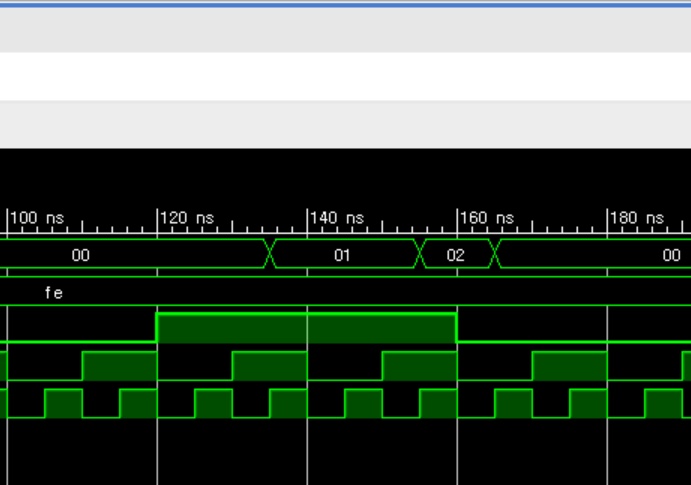
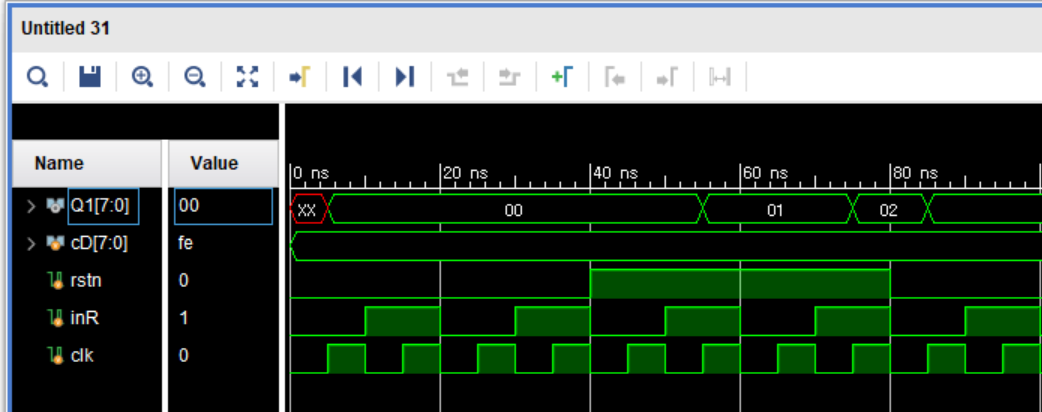
사용하고

Sin = 1b’0과

sD=8b’00001001은 shiftReg1, 2, 3을

위해 사용합니다.

1. Testbench waveform 설명
   * + Counter1



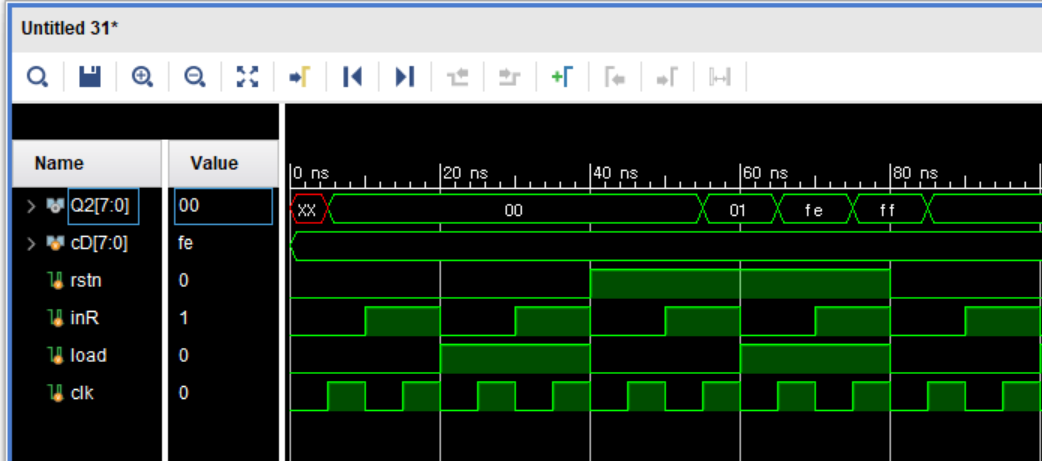
처음 5ns에서 clk값이 증가할 때, rstn의 값이 0이여서 Q값이 0으로 초기화 됩니다.

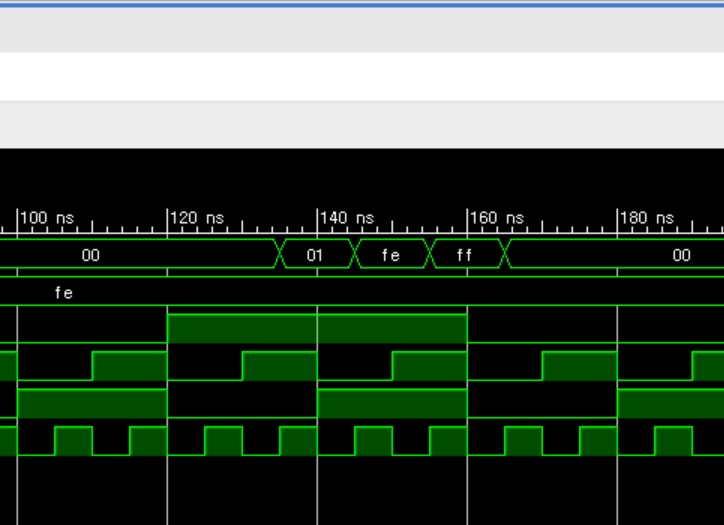
40ns까지는 rstn의 값이 계속 0이기에 inR의 신호가 1이여도 Q의 값은 증가하지 않습니다.(우선순위)

40ns에서 rstn의 값이 1이고 inR의 값이 0인 경우에는 변화가 없다가 inR이 1의 값이고 clk가 증가하는 경우(55ns)에는 Q의 값이 증가하는 모습을 확인할 수 있습니다.

이후 2까지 증가하다가 80ns에서 rstn의 값이 다시 0이 되므로 85ns에서 clk가 증가하는 경우 값이 00으로 초기화 되는 것을 확인할 수 있습니다.

* + - Counter2





처음 5ns에서 clk값이 증가할 때, rstn의 값이 0이여서 Q값이 0으로 초기화 됩니다.

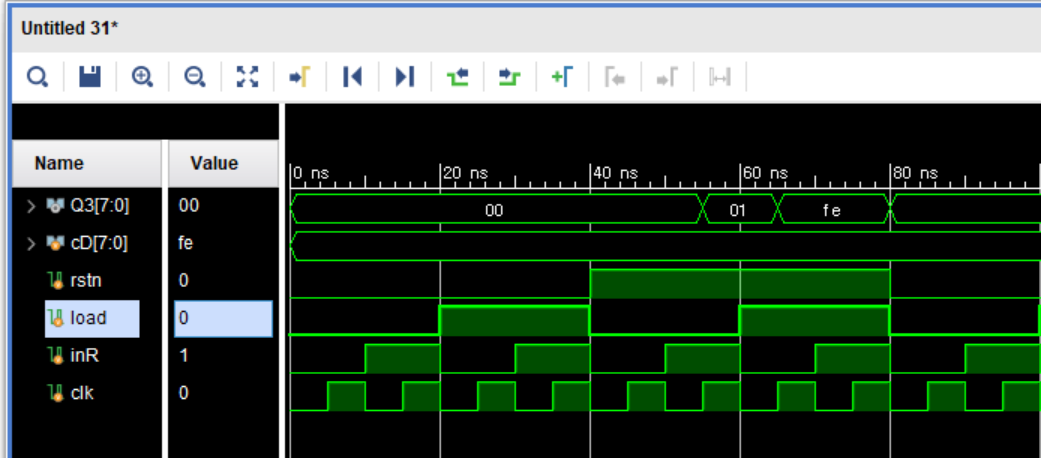
40ns까지는 rstn의 값이 계속 0이기에 inR의 신호와 load신호가 1이여도 Q의 값은 증가하지 않습니다.(우선순위)

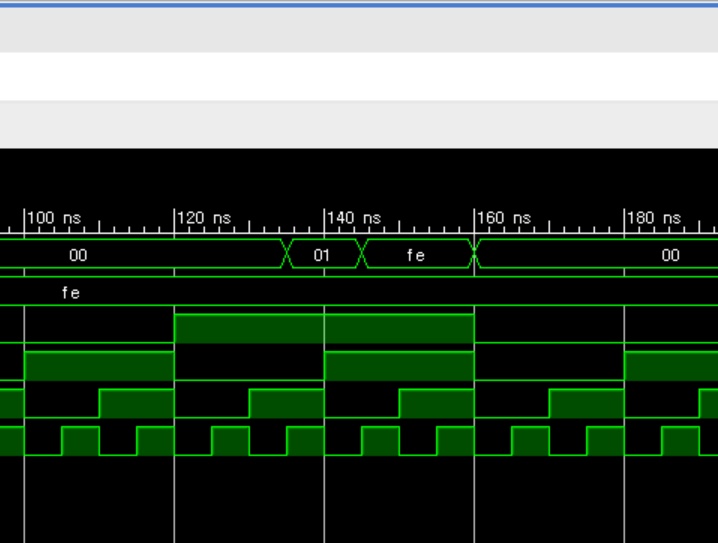
40ns에서 rstn의 값이 1이고 inR의 값이 0인 경우에는 변화가 없다가 inR이 1의 값이고 clk가 증가하는 경우(55ns)에는 Q의 값이 증가하는 모습을 확인할 수 있습니다.

60ns에서 inR의 값이 0, load의 값이 1로 바뀌게 되고 65ns에서 clk가 증가하는 경우, load가 작동하여 input값으로 주어졌던 fe(254)가 출력값으로 나오는 것을 확인할 수 있습니다.

이후 fe에서 inR로 인해 1씩 증가하다가 80ns에서 rstn의 값이 다시 0이 되므로 85ns에서 clk가 증가하는 경우 값이 00으로 초기화 되는 것을 확인할 수 있습니다.

* + - Counter3





처음 rstn의 값이 0이여서 Q값이 0으로 초기화 됩니다.

40ns까지는 rstn의 값이 계속 0이기에 load신호와 inR의 신호가 1이여도 Q의 값은 증가하지 않습니다.(우선순위)

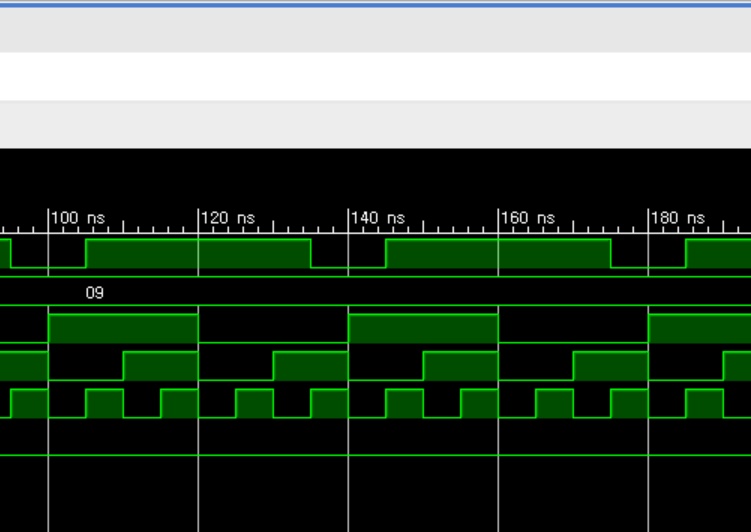
40ns에서 rstn의 값이 1이고 load의 값이 0인 경우에는 변화가 없다가 inR이 1의 값이고 clk가 증가하는 경우(55ns)에는 Q의 값이 1로 증가하는 모습을 확인할 수 있습니다.

60ns에서 inR의 값이 1이고 65ns에서 clk가 증가하는 경우에는 load가 작동하여 input값으로 주어졌던 fe(254)가 출력값으로 나오는 것을 확인할 수 있습니다.

이후 80ns에서 rstn의 값이 다시 0이 되므로 85ns에서 clk가 증가하는 경우 값이 00으로 초기화 되는 것을 확인할 수 있습니다.

* + - shiftReg1





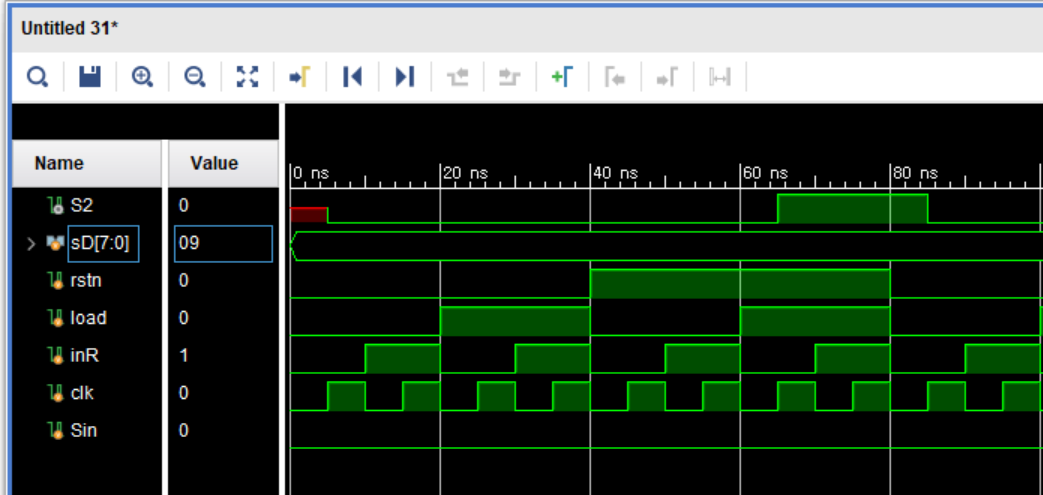
처음에는 입력값을 불러오지 않았기에 값을 모르므로 X값이 나타나게 됩니다.

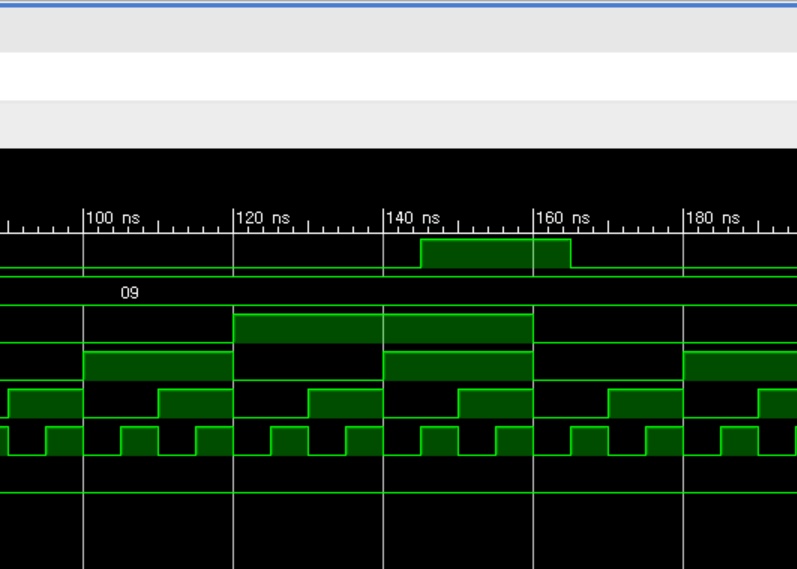
20ns에서 load의 값이 1이 되고 25ns에서 clk가 증가할 때, load가 작동하여 input값으로 주어졌던 9(00001001)에 의해 Sout(=Q[0]=1)이 1로 출력되는 것을 확인할 수 있습니다.

이때 inR의 신호가 바뀌어도 우선순위 때문에 출력값에 영향을 주지 못합니다.

55ns에서 load의 값이 0이고 inR의 값이 1이며 동시에 clk가 증가하는 경우에는 shiftR이 작동하여 오른쪽에서 한칸씩 미뤄지고 빈자리가 Sin(0)으로 채워져 Q(00010010)가 됩니다. 이로 인해 Sout이 0으로 출력됨을 확인할 수 있습니다.

* + - shiftReg2





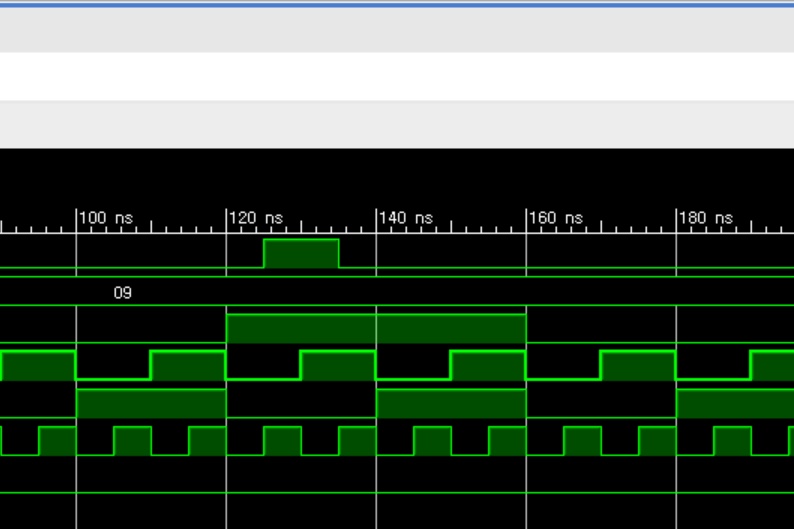
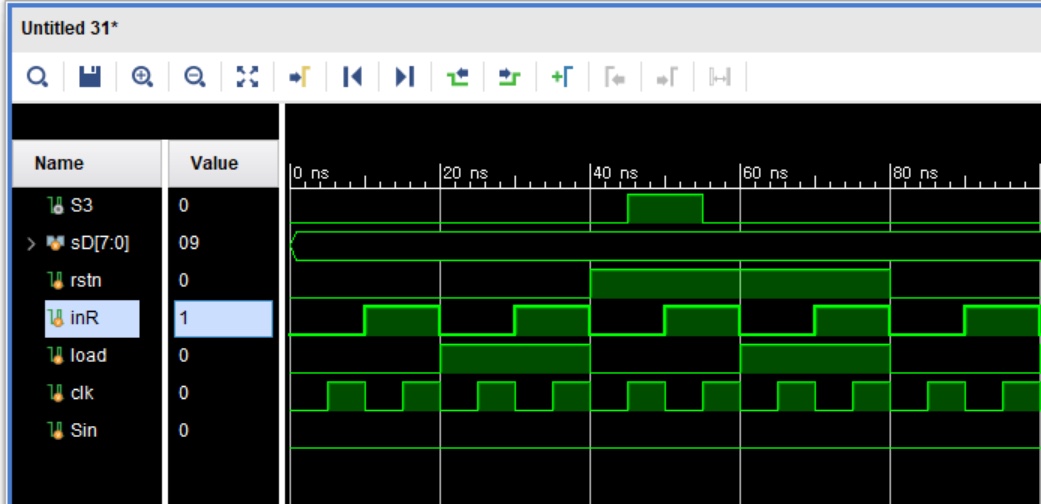
처음에는 입력값을 불러오지 않아값을 모르기에 X값이 나타나게 되지만 5ns에서 rstn의 값이 0이면서 clk값이 증가하는 경우, Q의 값으로 0이 출력됩니다.

40ns까지는 rstn의 값이 계속 0이기에 load신호와 inR의 신호가 1이여도 Q의 값은 증가하지 않습니다.(우선순위)

40ns에서 rstn의 값이 1이고 load의 값이 1인 경우에 load가 작동하여 input값으로 주어졌던 9(00001001)에 의해 Sout(=Q[0]=1)이 1로 출력되는 것을 확인할 수 있습니다.

이후 80ns에서 rstn의 값이 다시 0이 되므로 85ns에서 clk가 증가하는 경우 값이 00으로 초기화 되는 것을 확인할 수 있습니다.

* + - shiftReg3



rstn의 값이 0이여서 초기 Q값이 0으로 초기화 됩니다.

40ns까지는 rstn의 값이 계속 0이기에 inR과 load(shiftL)의 신호가 1이여도 Q의 값은 증가하지 않습니다.(우선순위)

40ns에서 rstn의 값이 1이고 inR의 값과 load(shiftL)의 값이 0인 경우에는 else문이 작동하여 입력값(00001001)이 들어가 출력값 1이 나오게 됩니다.

50ns에서 inR이 1이 되고 55ns에서 clk이 증가하여 shiftR이 작동하고 오른쪽에서 한칸씩 미뤄지고 빈자리가 Sin(0)으로 채워져 Q(00010010)가 됩니다. 이로 인해 Sout이 0으로 출력됨을 확인할 수 있습니다.

60ns에서 inR의 값이 0, load(shiftL)의 값이 1이 되어 shiftL이 작동하고 왼쪽에서 한칸씩 미뤄지고 빈자리가 Sin(0)으로 채워져 Q(00010010)가 됩니다. 이로 인해 Sout이 0으로 출력됨을 확인할 수 있습니다.

80ns에서 rstn의 값이 다시 0이 되므로 85ns에서 clk가 증가하는 경우 값이 00으로 초기화 되는 것을 확인할 수 있습니다.

* + - 과제를 통해 느낀점

동기식 reset와 비동기식 reset, 그리고 제어 신호의 우선순위에 따라 waveform이 다르게 나타나는 모습을 확인 할 수 있었습니다.