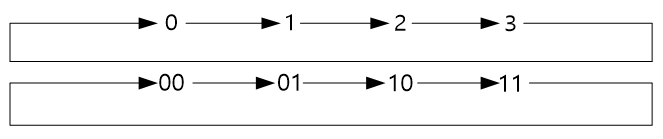
1. 12주차 결과보고서

|  |
| --- |
| - 2-bit counter  - 4-bit decade counter  - 4-bit 2421 decade counter  - 결과 검토 및 논의 사항  - 추가 이론 |

**20141196 김성희**

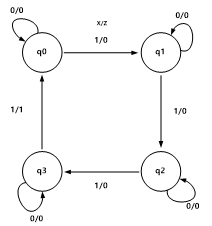
2-bit counter

**1. 논리 구조**



0부터 3까지 반복해서 카운팅하는 회로. 11 다음 수는 00으로, reset이 발생한다.  
4개의 상태가 필요하므로 2-bit 수가 필요하다.

**2. 상태도**

  
q0 = 00, q1 = 01, q2 = 10, q3 = 11을 의미하며 x/z = input과 output을 의미한다.  
input이 1이면 증가를, 0이면 같은 상태를 가리킨다.  
output은 오직 상태가 11에서 00으로 넘어갈 때 발생한다.

**3. 상태표**

<직관적 상태표>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **현재 상태** | **다음 상태** | | **Output** | |
| **X = 0** | **X = 1** | **X = 0** | **X = 1** |
| q0 | q0 | q1 | 0 | 0 |
| q1 | q1 | q2 | 0 | 0 |
| q2 | q2 | q3 | 0 | 0 |
| q3 | q3 | q0 | 0 | 1 |

\*간단한 구조이므로 D플립플롭을 이용하기로 한다.

<상세한 상태표>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **현재 상태** | **다음 상태** | | **Output (Z)** | | **D1 D2** | |
| **X = 0** | **X = 1** | **X = 0** | **X = 1** | **X = 0** | **X = 1** |
| 00 | 00 | 01 | 0 | 0 | 00 | 01 |
| 01 | 01 | 10 | 0 | 0 | 01 | 10 |
| 10 | 10 | 11 | 0 | 0 | 10 | 11 |
| 11 | 11 | 00 | 0 | 1 | 11 | 00 |

\*D플립플롭의 D는 q\* (next state)와 동일 한 값을 가지므로 ‘D1 D2’와 ‘다음 상태’ 가 동일하다.

**4. 카르노 맵**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  X | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |

D1 = Q1 (~X) + Q1 Q2 + (~Q1) Q2 X

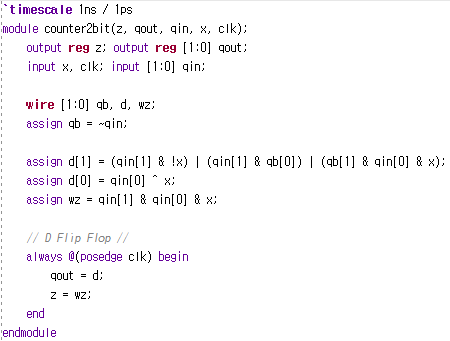
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  X | 00 | 01 | 11 | 10 |
| 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

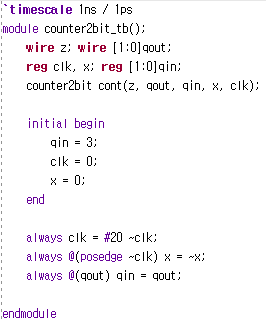
D2 = Q2 (~X) + (~Q2) X

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  X | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |

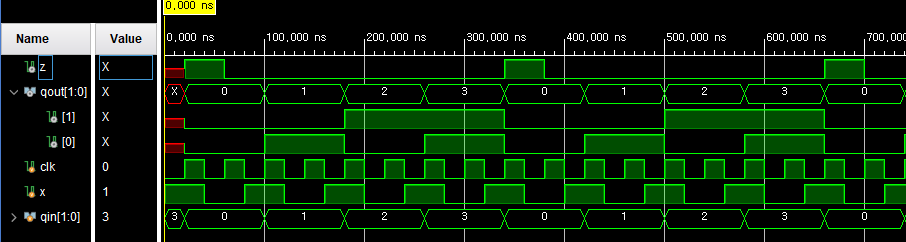
Z = Q1 Q2 X

**5. verilog code**

  
2-bit counter 모듈 코드다.  
위 4번의 카르노 맵에서 구한 D1, D2, Z는 코드에서 각각 d[1], d[0], wz를 의미한다.  
// D Flip Flop // 아래 부분은 rising edge triggered D Flip Flop을 구현한 부분으로, clock의 rising edge 상태일 때만 next state와 ouput z를 기록한다.

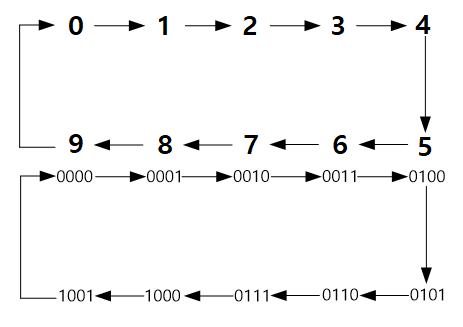
  
2-bit counter 모듈의 시뮬레이션을 위한 testbench 코드다.  
20ns 마다 clock을 바꿔주어 40ns의 주기를 가진 clock 신호를 생성하였고, x 값이 1일 때만 변한 다는 것을 보이기 위해 x값을 주기적으로 반전시켰다. qin에 qout을 넣은 이유는 현재 상태를 최신화하기 위함이다. 위 2-bit counter 모듈을 보면 다음 상태를 최신화하고 현재 상태는 그대로 남아있기 때문에 현재 상태를 최신화 할 필요가 있었다. qin = 3을 초기값으로 준 이유는 qout 즉 다음 상태가 00에서 시작할 수 있도록 하기 위함이다.

**6. simulation**

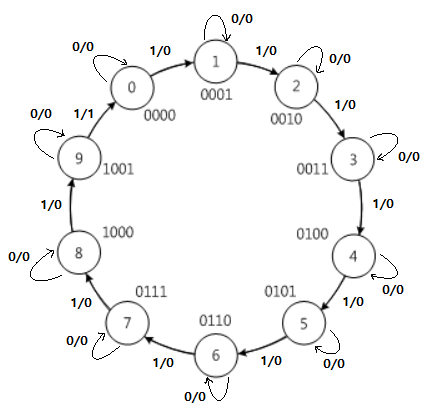
  
Name column의 변수들은 순서대로 z, qout[1:0], clk, x, qin[1:0]다.  
우선 qout 값이 0->1->2->3->0->1->… 처럼 계속 증가하고 3에서 0으로 reset되는 것을 볼 수 있으며, reset 될 때마다 z는 1의 값을 가짐으로써 reset되었음을 알려준다. (혹은 한 바퀴 돌았음을 알려준다.) qin이 qout과 같은 이유는 testbench 코드에서 qout이 바뀔 때마다 qin을 qout 값으로 최신화 해주었기 때문이다.

4-bit decade counter

**1. 논리 구조**

  
0~9까지 카운팅하는 회로. 9다음으로 0이 오며 이를 reset이라고 한다.  
10개의 상태가 필요하므로 4-bit 이진수를 사용하였다.

**2. 상태도**

  
input이 1일 때 state가 1 증가한다. state가 1001에서 0000이 되는 경우 output이 1이다. (나머진 0)

**3. 상태표**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **현재 상태** | **다음 상태** | | **Output** | |
| **X = 0** | **X = 1** | **X = 0** | **X = 1** |
| q0 | q0 | q1 | 0 | 0 |
| q1 | q1 | q2 | 0 | 0 |
| q2 | q2 | q3 | 0 | 0 |
| q3 | q3 | q4 | 0 | 0 |
| q4 | q4 | q5 | 0 | 0 |
| q5 | q5 | q6 | 0 | 0 |
| q6 | q6 | q7 | 0 | 0 |
| q7 | q7 | q8 | 0 | 0 |
| q8 | q8 | q9 | 0 | 0 |
| q9 | q9 | q0 | 0 | 1 |

qn = n, 즉 q0 = 0 = 0000, q4 = 4 = 0100, q8 = 8 = 1000

D플립플롭 사용. State를 4비트 이진수로 표현하므로 4개의 플립플롭 사용.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **현재 상태** | **다음 상태** | | **Output** | | **D1 D2 D3 D4** | |
| **X = 0** | **X = 1** | **X = 0** | **X = 1** | **X = 0** | **X = 1** |
| 0000 | 0000 | 0001 | 0 | 0 | 0000 | 0001 |
| 0001 | 0001 | 0010 | 0 | 0 | 0001 | 0010 |
| 0010 | 0010 | 0011 | 0 | 0 | 0010 | 0011 |
| 0011 | 0011 | 0100 | 0 | 0 | 0011 | 0100 |
| 0100 | 0100 | 0101 | 0 | 0 | 0100 | 0101 |
| 0101 | 0101 | 0110 | 0 | 0 | 0101 | 0110 |
| 0110 | 0110 | 0111 | 0 | 0 | 0110 | 0111 |
| 0111 | 0111 | 1000 | 0 | 0 | 0111 | 1000 |
| 1000 | 1000 | 1001 | 0 | 0 | 1000 | 1001 |
| 1001 | 1001 | 0000 | 0 | 1 | 1001 | 0000 |

D플립플롭의 식은 Q\* = D (Q\*는 next state) 이므로 ‘다음 상태’와 ‘D1 D2 D3 D4’가 동일하다.

**4. 카르노맵**

위 X = 0, 아래 X = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | X | 1 |
| 01 | 0 | 0 | X | 1 |
| 11 | 0 | 0 | X | X |
| 10 | 0 | 0 | X | X |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | X | 1 |
| 01 | 0 | 0 | X | 0 |
| 11 | 0 | 1 | X | X |
| 10 | 0 | 0 | X | X |

D1 = Q1 (~X) + Q1 (~Q4) + Q2 Q3 Q4 X

위 X = 0, 아래 X = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | X | 0 |
| 01 | 0 | 1 | X | 0 |
| 11 | 0 | 1 | X | X |
| 10 | 0 | 1 | X | X |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | X | 0 |
| 01 | 0 | 1 | X | 0 |
| 11 | 1 | 0 | X | X |
| 10 | 0 | 1 | X | X |

D2 = Q2 (~X) + Q2 (~Q3) + Q2 (~Q4) + (~Q2) Q3 Q4 X

위 X = 0, 아래 X = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | X | 0 |
| 01 | 0 | 0 | X | 0 |
| 11 | 1 | 1 | X | X |
| 10 | 1 | 1 | X | X |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | X | 0 |
| 01 | 1 | 1 | X | 0 |
| 11 | 0 | 0 | X | X |
| 10 | 1 | 1 | X | X |

D3 = Q3 (~X) + Q3 (~Q4) + (~Q1) (~Q3) Q4 X

위 X = 0, 아래 X = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | X | 0 |
| 01 | 1 | 1 | X | 1 |
| 11 | 1 | 1 | X | X |
| 10 | 0 | 0 | X | X |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | X | 1 |
| 01 | 0 | 0 | X | 0 |
| 11 | 0 | 0 | X | X |
| 10 | 1 | 1 | X | X |

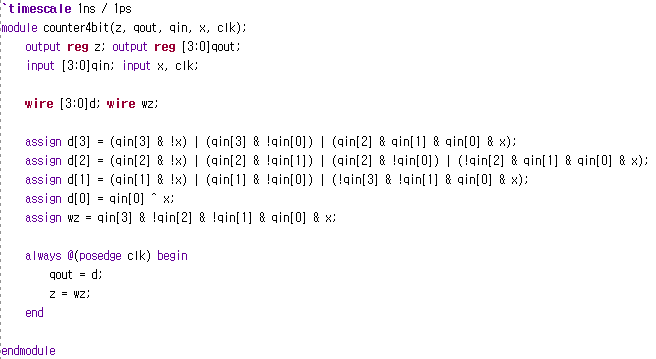
D4 = Q4 (~X) + (~Q4) X

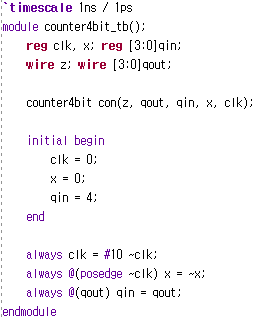
위 X = 0, 아래 X = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | X | 0 |
| 01 | 0 | 0 | X | 0 |
| 11 | 0 | 0 | X | X |
| 10 | 0 | 0 | X | X |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | X | 0 |
| 01 | 0 | 0 | X | 1 |
| 11 | 0 | 0 | X | X |
| 10 | 0 | 0 | X | X |

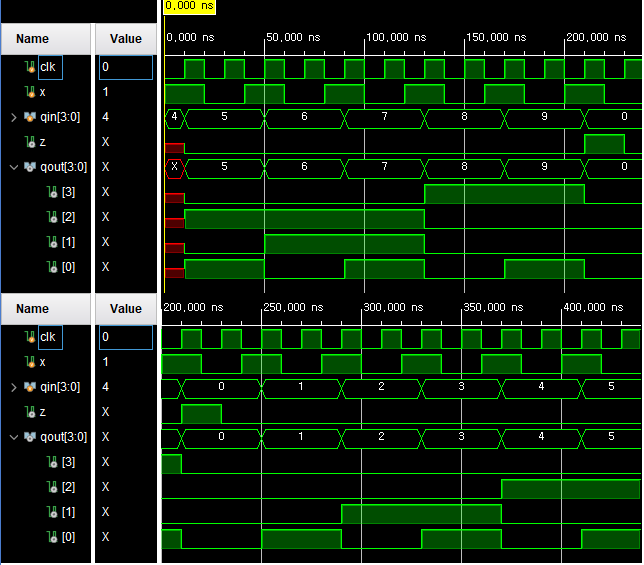
Z = Q1 (~Q2) (~Q3) Q4 X

**5. Verilog code**

  
4bit counter 모듈  
d[3] d[2] d[1] d[0] 는 각각 D1 D2 D3 D4를 의미하고 qin(qout)[3] qin(qout)[2] qin(qout)[1] qin(qout)[0]는 각각 Q1 Q2 Q3 Q4를 의미한다. always @(posedge clk) begin ~ end는 D 플립플롭을 재현한 코드다.

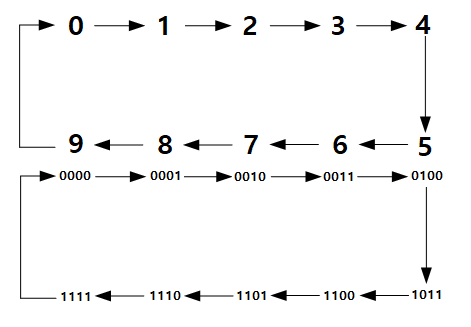
4bit counter testbench 모듈  
clock 주기는 20ns이고, always @(qout)~~~ 코드는 현재 상태를 최신화 하는 코드다.

**6. simulation**

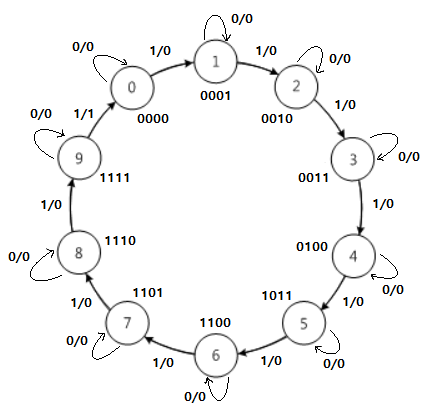
  
(잘 보이도록 2개로 쪼개서 올림, 위와 아래 그림이 같은 시뮬레이션이라고 보면 된다.)  
clk가 rising edge 상태이고 x가 1일 때 qout이 바뀌는 것을 볼 수 있다.  
qout을 보면 5->6-> … -> 9 -> 0 -> 1 -> … -> 5처럼 증가하는 것을 볼 수 있는데 9 다음 숫자는 0으로 reset되는 것을 볼 수 있다. qin이 qout과 같은 값을 가지는 이유는 qin을 통해 qout을 구한 뒤에 곧 바로 qin을 qout으로 최신화 시키기 때문이다. z는 평소에 0이다가 9에서 0으로 reset 될 때 1을 가진다.

1. 4-bit 2421 decade counter

**1. 논리 구조**



**2. 상태도**

  
4-bit decade counter와 동일. 5~9의 경우 0101~1001 대신 1011~1111을 이용. 즉 5이상의 수는 +6을 해주는 것과 동일하다.

**3. 상태표**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **현재 상태** | **다음 상태** | | **Output** | |
| **X = 0** | **X = 1** | **X = 0** | **X = 1** |
| q0 | q0 | q1 | 0 | 0 |
| q1 | q1 | q2 | 0 | 0 |
| q2 | q2 | q3 | 0 | 0 |
| q3 | q3 | q4 | 0 | 0 |
| q4 | q4 | q5 | 0 | 0 |
| q5 | q5 | q6 | 0 | 0 |
| q6 | q6 | q7 | 0 | 0 |
| q7 | q7 | q8 | 0 | 0 |
| q8 | q8 | q9 | 0 | 0 |
| q9 | q9 | q0 | 0 | 1 |

D플립플롭 사용. State를 4비트 이진수로 표현하므로 4개의 플립플롭 사용.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **현재 상태** | **다음 상태** | | **Output** | | **D1 D2 D3 D4** | |
| **X = 0** | **X = 1** | **X = 0** | **X = 1** | **X = 0** | **X = 1** |
| 0000 | 0000 | 0001 | 0 | 0 | 0000 | 0001 |
| 0001 | 0001 | 0010 | 0 | 0 | 0001 | 0010 |
| 0010 | 0010 | 0011 | 0 | 0 | 0010 | 0011 |
| 0011 | 0011 | 0100 | 0 | 0 | 0011 | 0100 |
| 0100 | 0100 | 1011 | 0 | 0 | 0100 | 1011 |
| 0101 | 1011 | 1100 | 0 | 0 | 1011 | 1100 |
| 0110 | 1100 | 1101 | 0 | 0 | 1100 | 1101 |
| 0111 | 1101 | 1110 | 0 | 0 | 1101 | 1110 |
| 1000 | 1110 | 1111 | 0 | 0 | 1110 | 1111 |
| 1001 | 1111 | 0000 | 0 | 1 | 1111 | 0000 |

D플립플롭의 식은 Q\* = D (Q\*는 next state) 이므로 ‘다음 상태’와 ‘D1 D2 D3 D4’가 동일하다

**4. 카르노맵**

위 X = 0, 아래 X = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 1 | X |
| 01 | 0 | X | 1 | X |
| 11 | 0 | X | 1 | 1 |
| 10 | 0 | X | 1 | X |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 1 | X |
| 01 | 0 | X | 1 | X |
| 11 | 0 | X | 0 | 1 |
| 10 | 0 | X | 1 | X |

D1 = Q1 (~X) + Q1 (~Q3) + Q1 (~Q4) + Q1 (~Q2) + Q2 (~Q3) X

위 X = 0, 아래 X = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 1 | X |
| 01 | 0 | X | 1 | X |
| 11 | 0 | X | 1 | 0 |
| 10 | 0 | X | 1 | X |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 1 | X |
| 01 | 0 | X | 1 | X |
| 11 | 1 | X | 0 | 1 |
| 10 | 0 | X | 1 | X |

D2 = Q2 (~X) + Q1 (~Q3) + Q1 (~Q4) + (~Q2) Q3 Q4 X

위 X = 0, 아래 X = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | X |
| 01 | 0 | X | 0 | X |
| 11 | 1 | X | 1 | 1 |
| 10 | 1 | X | 1 | X |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 0 | X |
| 01 | 1 | X | 1 | X |
| 11 | 0 | X | 0 | 0 |
| 10 | 1 | X | 1 | X |

D3 = Q3 (~X) + Q3 (~Q4) + (~Q3) Q4 X + (~Q1) Q2 X

위 X = 0, 아래 X = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | X |
| 01 | 1 | X | 1 | X |
| 11 | 1 | X | 1 | 1 |
| 10 | 0 | X | 0 | X |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 1 | X |
| 01 | 0 | X | 0 | X |
| 11 | 0 | X | 0 | 0 |
| 10 | 1 | X | 1 | X |

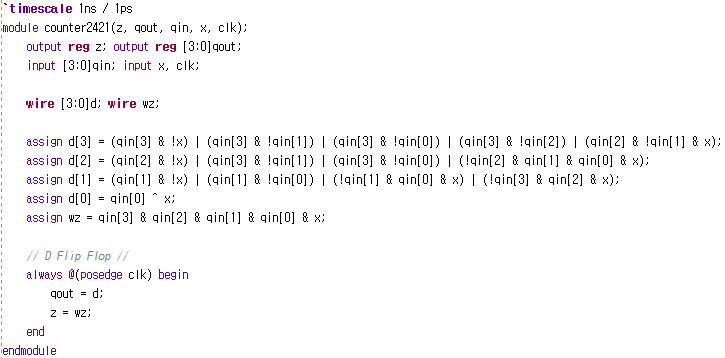
D4 = Q4 (~X) + (~Q4) X

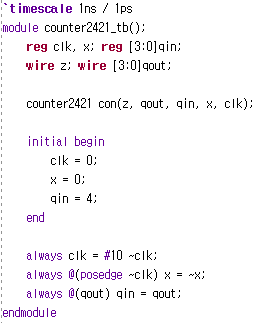
위 X = 0, 아래 X = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | X |
| 01 | 0 | X | 0 | X |
| 11 | 0 | X | 0 | 0 |
| 10 | 0 | X | 0 | X |
| Q1 Q2  Q3 Q4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | X |
| 01 | 0 | X | 0 | X |
| 11 | 0 | X | 1 | 0 |
| 10 | 0 | X | 0 | X |

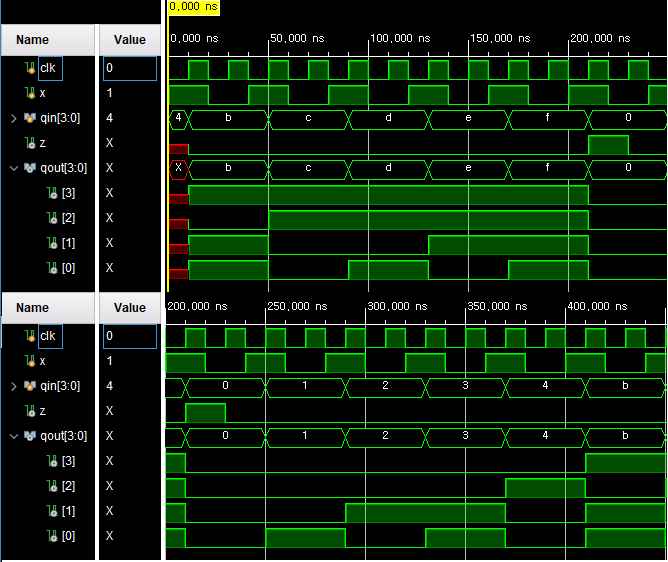
Z = Q1 Q2 Q3 Q4 X

**5. Verilog code**

  
4-bit 2421 decade counter 모듈 코드  
d[3] ~ [0]는 각각 D1 ~ 4에 해당하고, qin(qout)[3] ~[0]는 각각 Q1 ~ 4에 해당한다. D Flip Flop part에는 rising edge trigger 방식의 플립플롭을 구현했다. wz는 z에 output을 전달해 줄 wire 변수다.

  
이전 실험에서의 counter4bit\_tb 모듈 코드와 동일하다.

**6. simulation**

  
4-bit decade counter에서 simulation 그림과 마찬가지로 잘 보이도록 쪼개어 상/하로 나누었다.  
qout의 값 순차적으로 b c d e f 0 1 2 3 4 b을 가지는데, 여기서 b~f는 각각 11~15를 의미한다. 2421 code는 0~4까지는 그대로지만 5~9까지는 +6을 한 값을 가진다. 따라서 b c d e f 0 1 2 3 4 b의 의미는 5 6 7 8 9 0 1 2 3 4 5이다.

기타 논의 및 검토 사항 + 추가 이론

- 입력 값 x가 존재하지 않아도 clock 주기를 조정하면 원하는 시간 간격으로 카운트를 할 수 있지 않았을까 싶다. 카르노맵도 5개의 변수에서 4개의 변수로 변하면서 좀 더 쉬운 형태를 띄었을 것이다.

- 2421 code의 경우 0 1 2 3 4 5 6 7 9 를 0 1 2 3 4 b c d e f로 나타낼 수 있지만 다음과 같이 나타낼 수도 있다. 0~4는 그대로 0~4, 5~9는 4~0에 대한 1의 보수. 예를 들어 7은 d로 대신해서 쓰는데, d = 1101로, 2 = 0010에 대한 1의 보수이다. 따라서 2421 코드는 a+b = 9일 때, a와 b가 서로 1의 보수 관계에 있다. 다음 표를 보면 쉽게 이해할 수 있다.

|  |  |  |  |
| --- | --- | --- | --- |
| **십진수** | **8421 코드** | **1의 보수 🡪** | **2421 코드** |
| 0 | 0000 | 9(1111)와  1의 보수관계 | 0000 |
| 1 | 0001 | 8(1110)와  1의 보수관계 | 0001 |
| 2 | 0010 | 7(1101)와  1의 보수관계 | 0010 |
| 3 | 0011 | 6(1100)와  1의 보수관계 | 0011 |
| 4 | 0100 | 5(1011)와  1의 보수관계 | 0100 |
| 5 | 0101 | 4(0100)와  1의 보수관계 | 1011 |
| 6 | 0110 | 3(0011)와  1의 보수관계 | 1100 |
| 7 | 0111 | 2(0010)와  1의 보수관계 | 1101 |
| 8 | 1000 | 1(0001)와  1의 보수관계 | 1110 |
| 9 | 1001 | 0(0000)와  1의 보수관계 | 1111 |