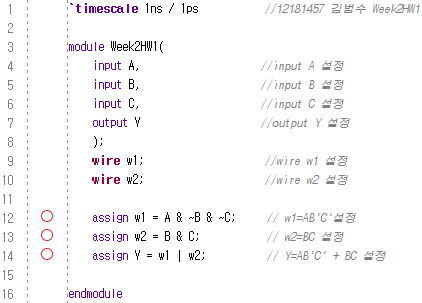
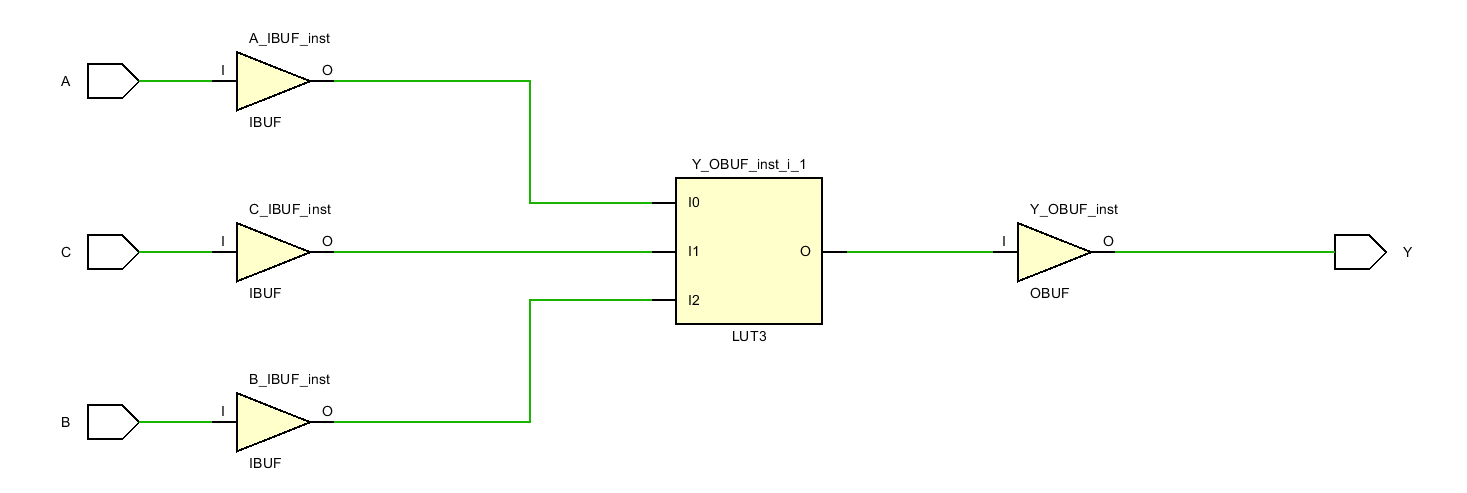
**Lab1.**

# **Homework1.Boolian expression**

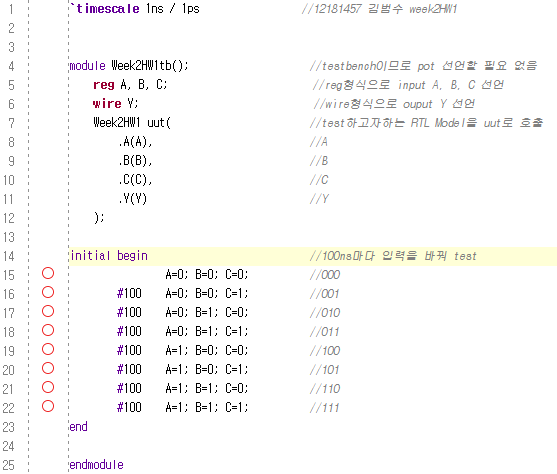
**Verilog Code / 주석**



**Synthesis Report**

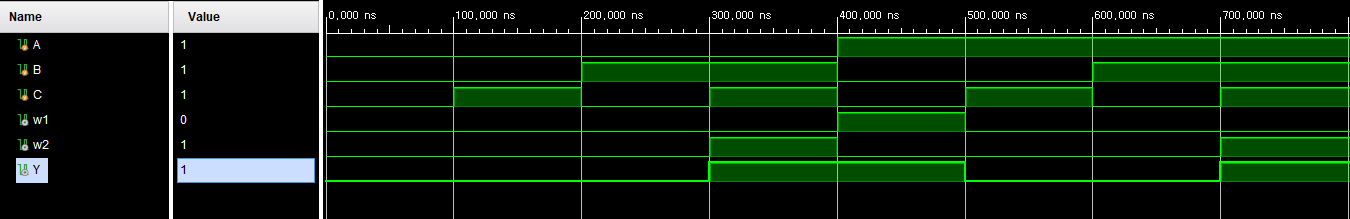
****

**Test Bench Code**

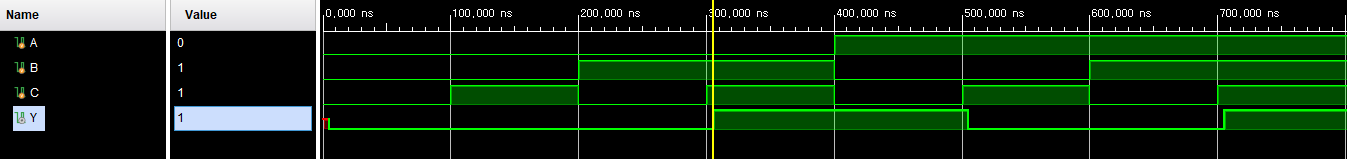


**Simulation Result**

Functional Simulation



Timing simulation

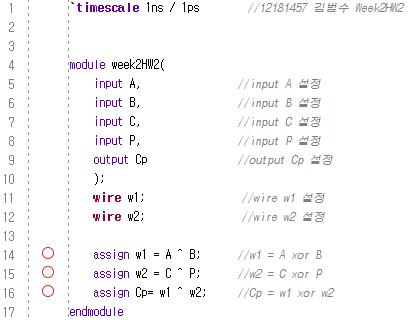


|  |
| --- |
| **Discussion** |
| . **- Verilog Coding을 시작하기 전 작성한 Block Diagram**    주어진 진리표를 살펴보면 A, B, C총 3개의 input을 받고 output으로 Y를 출력하는 것을 확인할 수 있다.  따라서 Block Diagram을 위와 같이 그렸다.  프로그램상 Block Diagram에는 각 input과 output에 버퍼가 추가되어 있는 것을 확인할 수 있다.  버퍼는 전기적으로 성질이 다른 두 회로 사이에 전기적으로 문제가 생기지 않도록 연결해주는 회로이다.  **- 작성한 Verilog Module 분석**    코드를 작성하기 앞서 진리표를 카르노맵으로 바꾸어 입력과 출력의 관계를 확인하였다.  Y=AB’C’+BC라는 식을 얻을 수 있었고 이를 logic gate로 표현하면 아래와 같았다.    이를 verilog로 구현하기 위하여 코드를 작성하였다.  코드는 먼저 input으로 A, B, C를 설정하고 output으로 Y를 설정하였다.  이후 wire로 w1, w2를 설정하였다.  먼저 w1=AB’C’와 w2=BC를 설정하기 위해 and를 사용하여 표현하였다.  이후 or를 사용하여 Y=AB’C’+BC=w1+w2로 설정하였다.  **-작성한 code가 하는 동작**  Input을 가하면 Y=AB’C’+BC가 되도록 동작한다.  **-실제 logic diagram**    앞서 분석한 결과를 토대로 시뮬레이션 상에서 logic diagram을 확인해 보았다.  simulation에서는 3개의 입력으로 w1을 출력하도록 한 code와 달리 2개의 입력이 하나의 gate를 거치도록 설계되어 2번의 gate를 거쳐 w1이 생성되도록 설계되어 있는 것을 확인할 수 있었다.  이를 통해 시뮬레이션 특성상 2개의 입력만을 받을 수 있는 gate로만 설계될 수 있다는 것을 알게 되었다.  **- 작성한 Test Bench Code 분석**  먼저 reg로 A, B, C를 선언하였고 wire로 Y를 선언하였다.  이 값을 출력하기 위하여 uut로 앞서 작성한 모듈을 불러왔다.  이후 000부터 100ns의 간격으로 111까지 입력을 변화시켰다.  **- Simulation 파형 분석**  먼저 Functional Simulation에서는 앞서 진리표를 카르노맵으로 바꿔 얻은 식과 동일한 결과를 얻었다.  하지만 Timing simulation에서는 output Y는 입력에 비해5.225ns의 delay가 나타났다.    **- 작성한 Code가 잘 동작하지 않는 이유**  위 사진을 통해 5.225ns의 delay가 발생하는 것을 확인할 수 있었다.  이 delay가 어디서 발생하는지 확인하기 위하여 Y=AB로 설정하여 시뮬레이션을 다시 실행하였다.    하지만 이때도 같은 5.225ns의 delay가 측정되었다.  AND gate를 2개와 OR Gate 1개를 지날 때와 AND Gate를 1개 지날 때 같은 delay가 측정된 것이다.  이로 보아 delay가 and Gate를 사용하기만 하면 최소 5.225ns의 delay를 나타내는 것 같다. |
|  |

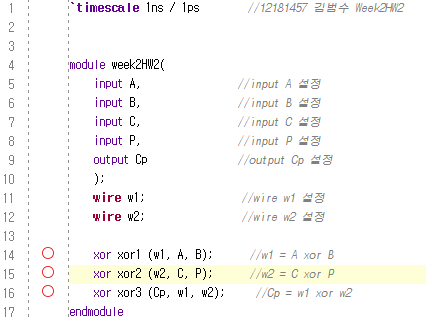
# **Homework2. 4bit even parity checker**

**Verilog Code / 주석**

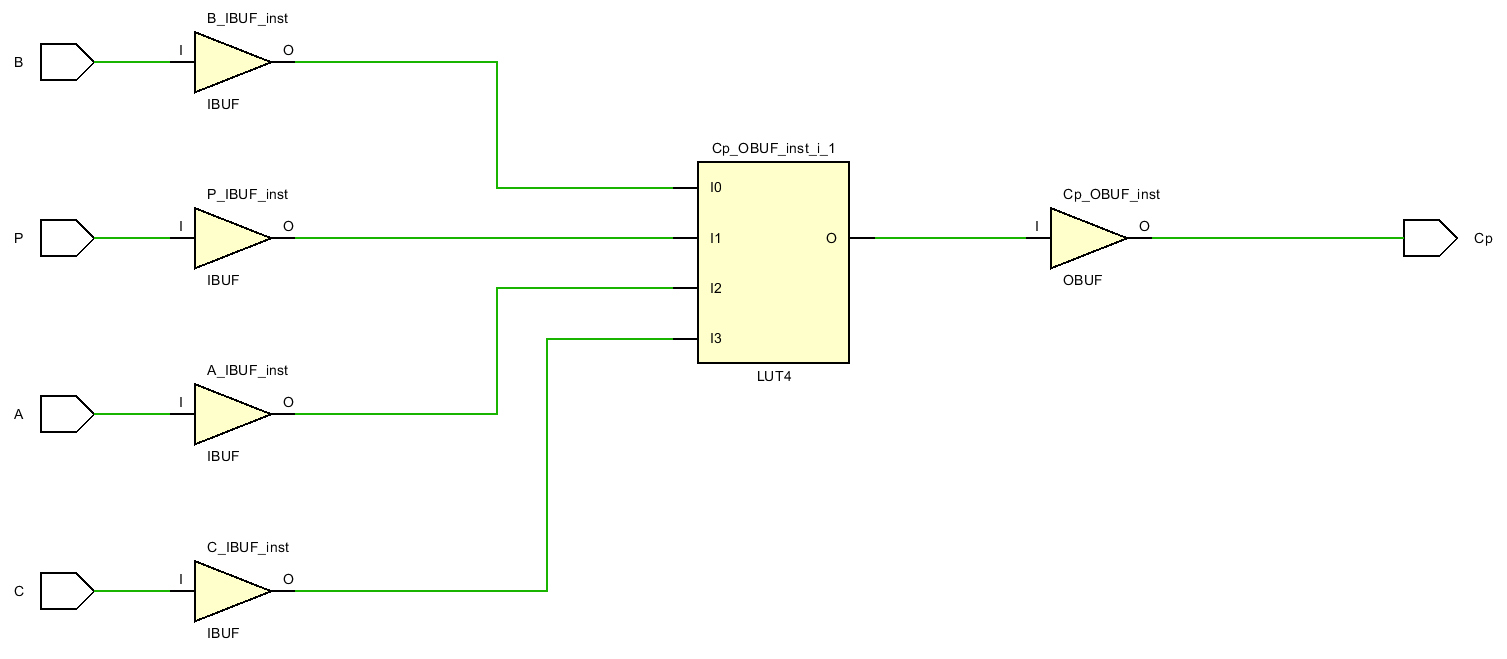
- Data Flow Model



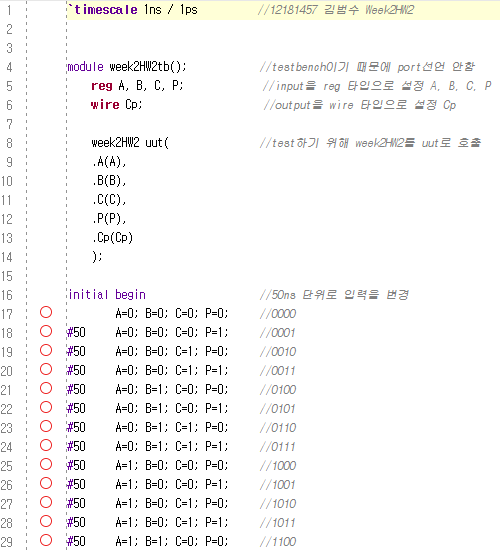
- Structural Modeling

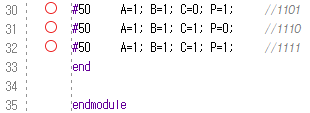


**Synthesis Report**



**Test Bench Code**

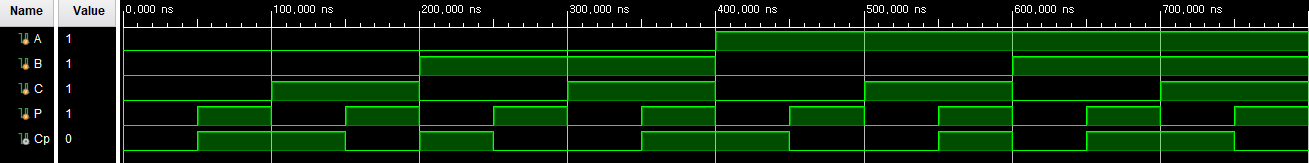




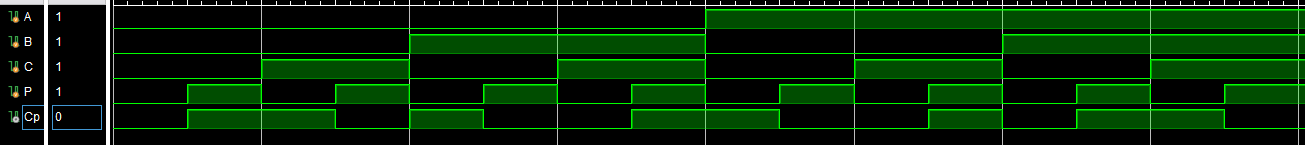
**Simulation Result**

Functional Simulation

- Data Flow Model

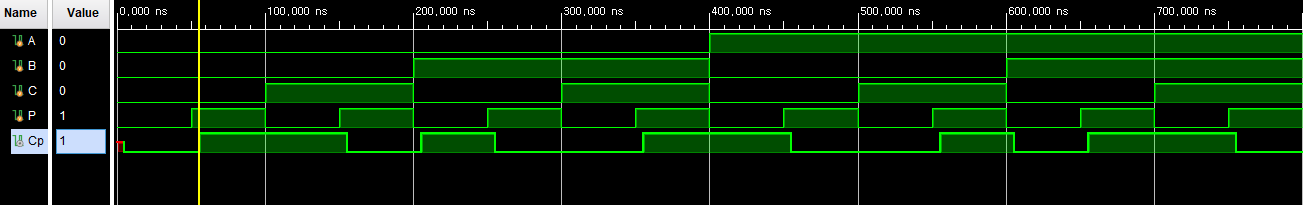


- Structural Modeling

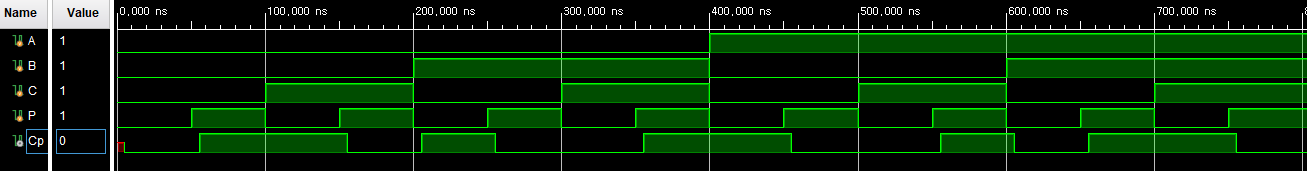


Timing simulation

- Data Flow Model



- Structural Modeling



|  |
| --- |
| **Discussion** |
| . **- Verilog Coding을 시작하기 전 작성한 Block Diagram**    주어진 진리표를 살펴보면 A, B, C, P총 4개의 input을 받고 output으로 CP를 출력하는 것을 확인할 수 있다.  따라서 Block Diagram을 위와 같이 그렸다.  프로그램상 Block Diagram에는 각 input과 output에 버퍼가 추가되어 있는 것을 확인할 수 있다.  버퍼는 전기적으로 성질이 다른 두 회로 사이에 전기적으로 문제가 생기지 않도록 연결해주는 회로이다.  - **작성한 Verilog Module 설명**    코드를 작성하기 앞서 진리표를 카르노맵으로 바꾸어 입력과 출력의 관계를 확인하였다.  카르노맵에서는 딱히 유효한 결과를 얻을 수 없었고 수식으로 정리하였다.  Y=A ⊕B ⊕C ⊕P라는 식을 얻을 수 있었고 이를 logic gate로 표현하면 아래와 같았다.    이를 verilog로 구현하기 위하여 코드를 작성하였다.  코드는 먼저 input으로 A, B, C, P를 설정하고 output으로 CP를 설정하였다.  이후 wire로 w1, w2를 설정하였다.  먼저 w1=A⊕B와 w2=C⊕P를 설정하기 위해 xor를 사용하여 표현하였다.  이후 xor를 사용하여Y=A ⊕B ⊕C ⊕P로 설정하였다.  Data Flow Modeling과 Structural Modeling 두 방법으로 모두 실행해 보았으나 모든 부분에서 동일한 모습을 보이는 것을 확인할 수 있었다.  **-작성한 code가 하는 동작**  4bit even parity checker는 1이 짝수 개일 경우에 0을 출력하고 홀수 개일 경우에 출력으로 1을 내보낸다.  위에서 작성한 코드에 따라 input을 가하게 되면 Y=A ⊕B ⊕C ⊕P에 맞춰 ouput이 생성되는 것을 확인할 수 있다.  **-실제 logic diagram**    앞서 분석한 결과를 토대로 시뮬레이션 상에서 logic diagram을 확인해 보았다.  Simulation과 수식을 정리하여 그린 회로가 동일한 것을 확인할 수 있었다.  또한 한 개의 Gate에 2개의 입력이 들어가는 것을 확인할 수 있었다.  **- 작성한 Test Bench Code 분석**  먼저 reg로 A, B, C, P를 선언하였고 wire로 CP를 선언하였다.  이 값을 출력하기 위하여 uut로 앞서 작성한 모듈을 불러왔다.  이후 0000부터 50ns의 간격으로 1111까지 입력을 변화시켰다.  **- Simulation 파형**  먼저 Functional Simulation에서는 앞서 진리표를 수식으로 정리해 얻은 식과 동일한 결과를 얻었다.  하지만 Timing simulation에서는 output CP는 입력에 비해5.349ns의 delay가 나타났다.    **- 작성한 Code가 잘 동작하지 않는 이유**  위 사진을 통해 5.349ns의 delay가 발생하는 것을 확인할 수 있었다.  이 delay가 어디서 발생하는지 확인하기 위하여 Y= A ⊕B ⊕C ⊕P⊕D 로 설정하여 시뮬레이션을 다시 실행하였다.  하지만 이때도 같은 5.349ns의 delay가 측정되었다.  XOR gate를 2개 지날 때와 XOR Gate 3개를 지날 때 같은 delay가 측정된 것이다.    이로 보아 delay가 XOR Gate를 사용하기만 하면 최소 5.349ns로 나타나는 것 같다. |
|  |