4주차 결과보고서

전공: 심리학과 학년: 3학년 학번: 20190345 이름: 김동현

**1.**

Verilog를 이용하여 NAND, NOR, XOR, AOI 논리 게이트를 구현하고, simulation을 통해 결과값을 예측한다. 이후 FPGA를 활용하여 회로를 검증한다.

**2.**

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림1. 4-input NAND gate design source code>

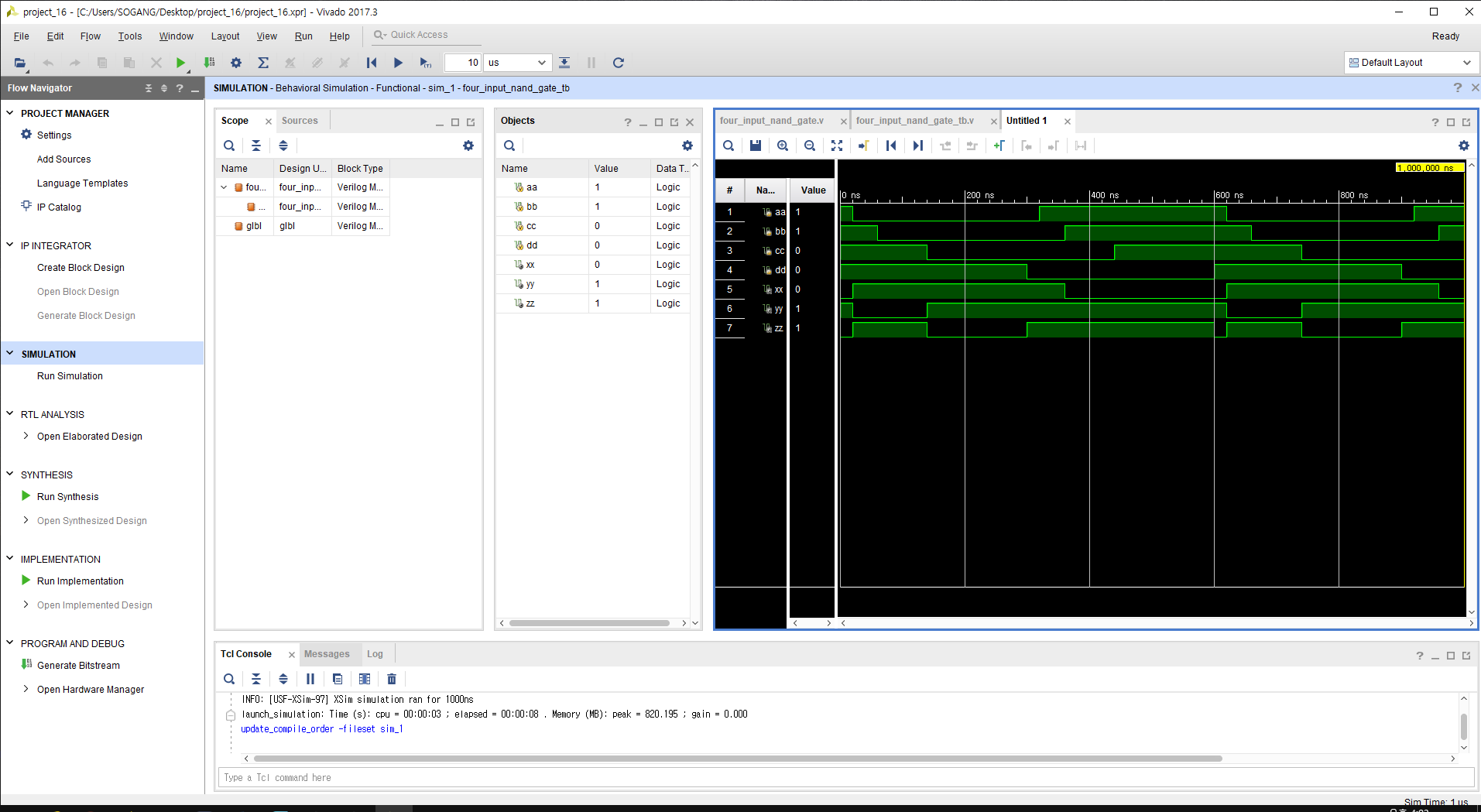
위 그림은 4개의 입력과 3개의 출력을 가지는 NAND gate의 디자인 소스 코드이다. 변수 x는 입력 a와 b의 NAND 연산을 한 결과인 ~(a & b)를 저장하고 있으며, 변수 y는 앞서 연산 결과인 x와 c의 NAND 연산 결과 ~(x & c)를, 변수 z는 앞선 연산 결과인 y와 d의 NAND 연산 결과 ~(y & d)를 저장하고 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림2. 4-input NAND gate simulation source code>

위 그림은 디자인 소스 코드에서 구현한 4\_input\_NAND\_gate를 시뮬레이션으로 구현하기 위한 시뮬레이션 코드이다. 입력값에 값을 할당하고, 값이 변경되는 주기와, 전체 시간을 설정하였다.



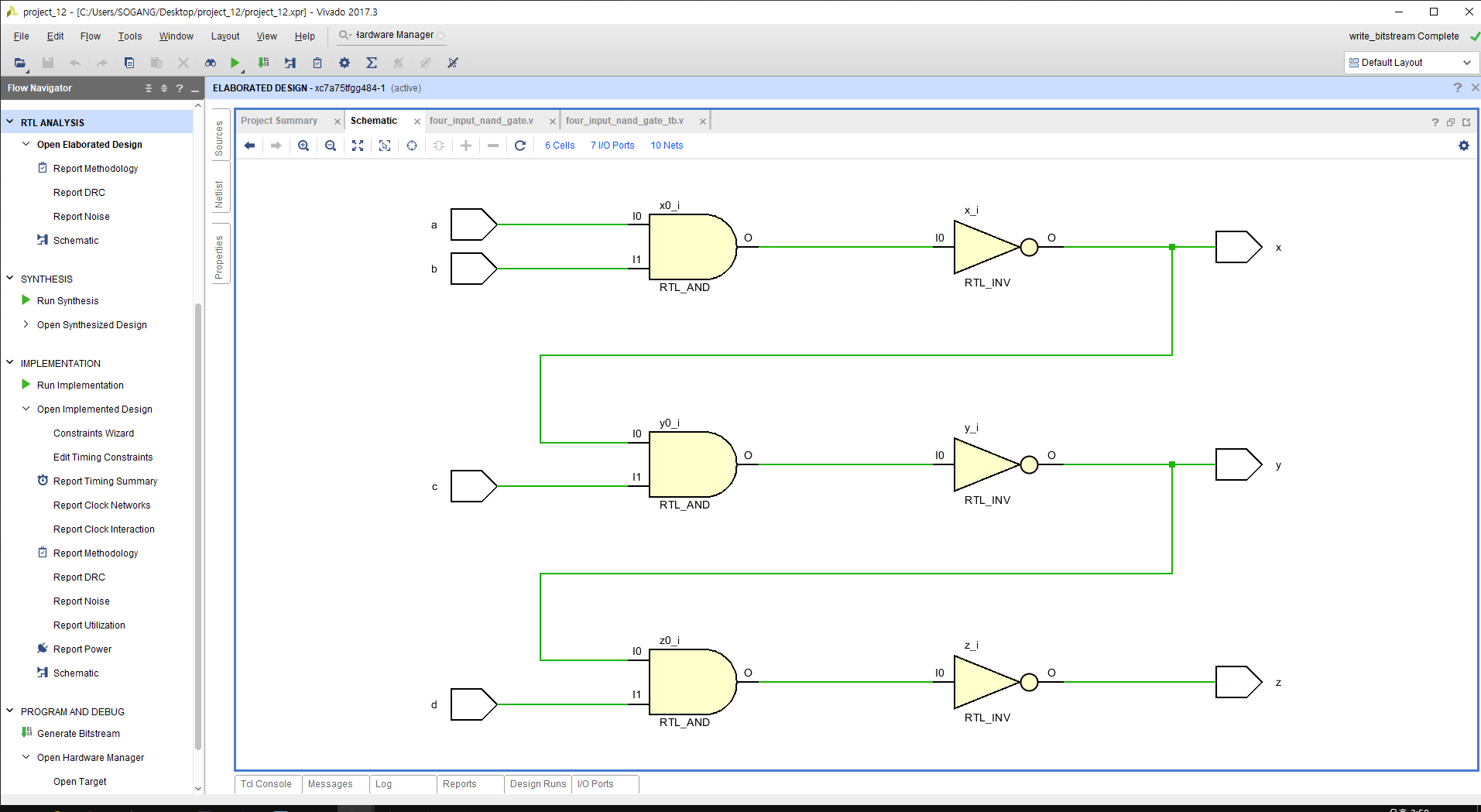
<그림3. 4-input NAND gate simulation>

시뮬레이션 소스 파일을 실행한 결과 다음과 같은 시뮬레이션 결과가 나왔다. NAND는 모든 입력이 1인 경우 0을, 나머지 경우에는 1을 출력한다. 따라서 xx는 aa와 bb가 모두 1인 경우, yy는 xx와 cc가 모두 1인 경우, zz는 yy와 dd가 모두 1인 경우 0을 출력하고, 나머지 경우에는 모두 1을 출력한다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input a | Input b | Input c | Input d | Output x | Output y | Output z |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |

<table1. 4-input NAND gate truth table>

NAND gate의 시뮬레이션 결과를 진리표로 정리하였다.



<그림4. 4-input NAND gate schematic>

위 그림은 NAND gate를 schematic형태로 구현한 모습이다. AND gate와 NOT gate를 지나며 NAND게이트가 됨을 알 수 있다.

**3.**

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림5. 4-input NOR gate design source code>

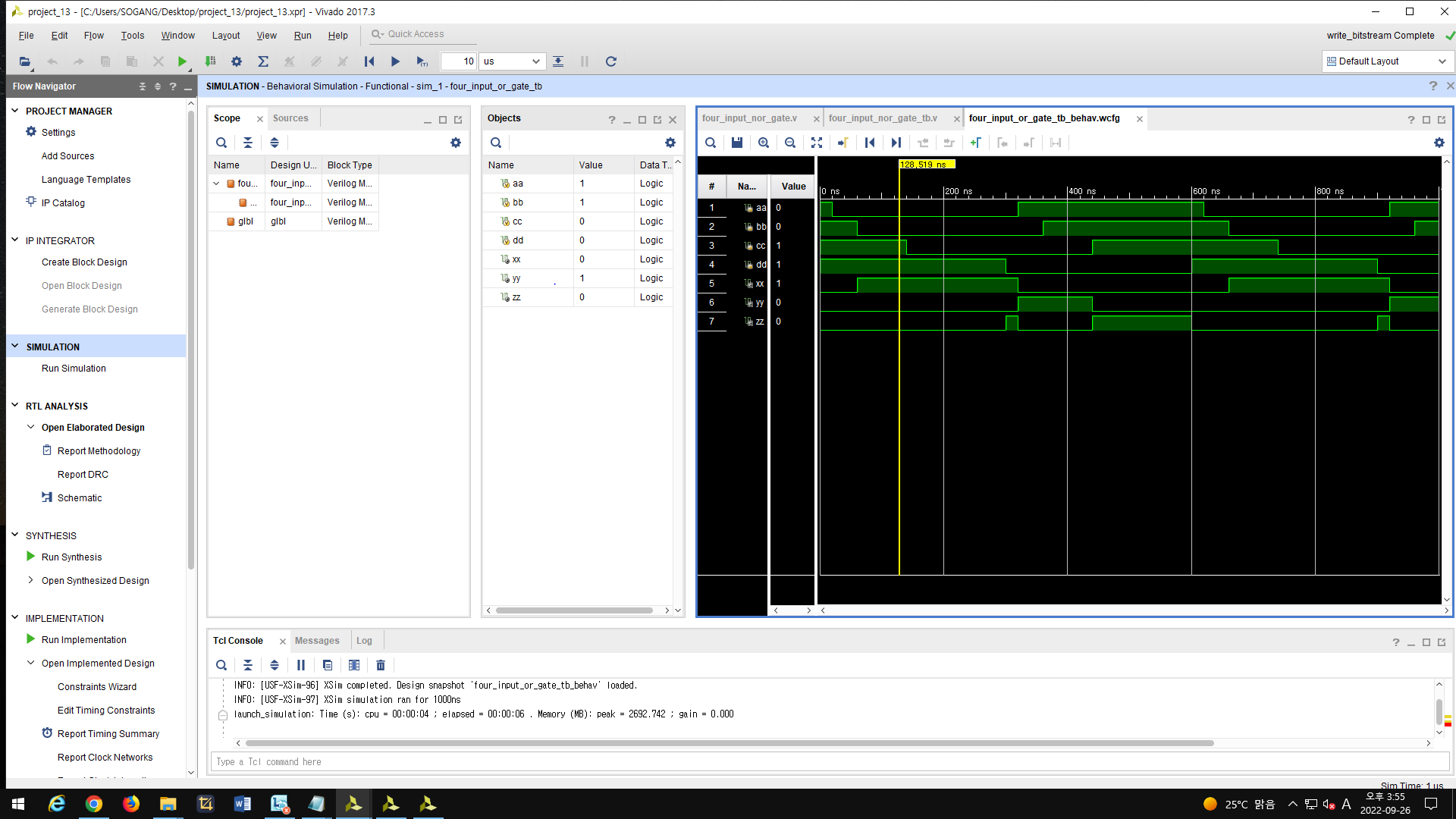
위 그림은 4개의 입력과 3개의 출력을 가지는 NOR gate의 디자인 소스 코드이다. 변수 x는 입력 a와 b의 NOR 연산을 한 결과인 ~(a | b)를 저장하고 있으며, 변수 y는 앞서 연산 결과인 x와 c의 NOR 연산 결과 ~(x | c)를, 변수 z는 앞선 연산 결과인 y와 d의 NOR 연산 결과 ~(y | d)를 저장하고 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림6. 4-input NOR gate simulation source code>

위 그림은 디자인 소스 코드에서 구현한 4\_input\_NOR\_gate를 시뮬레이션으로 구현하기 위한 시뮬레이션 코드이다. 입력값에 값을 할당하고, 값이 변경되는 주기와, 전체 시간을 설정하였다.



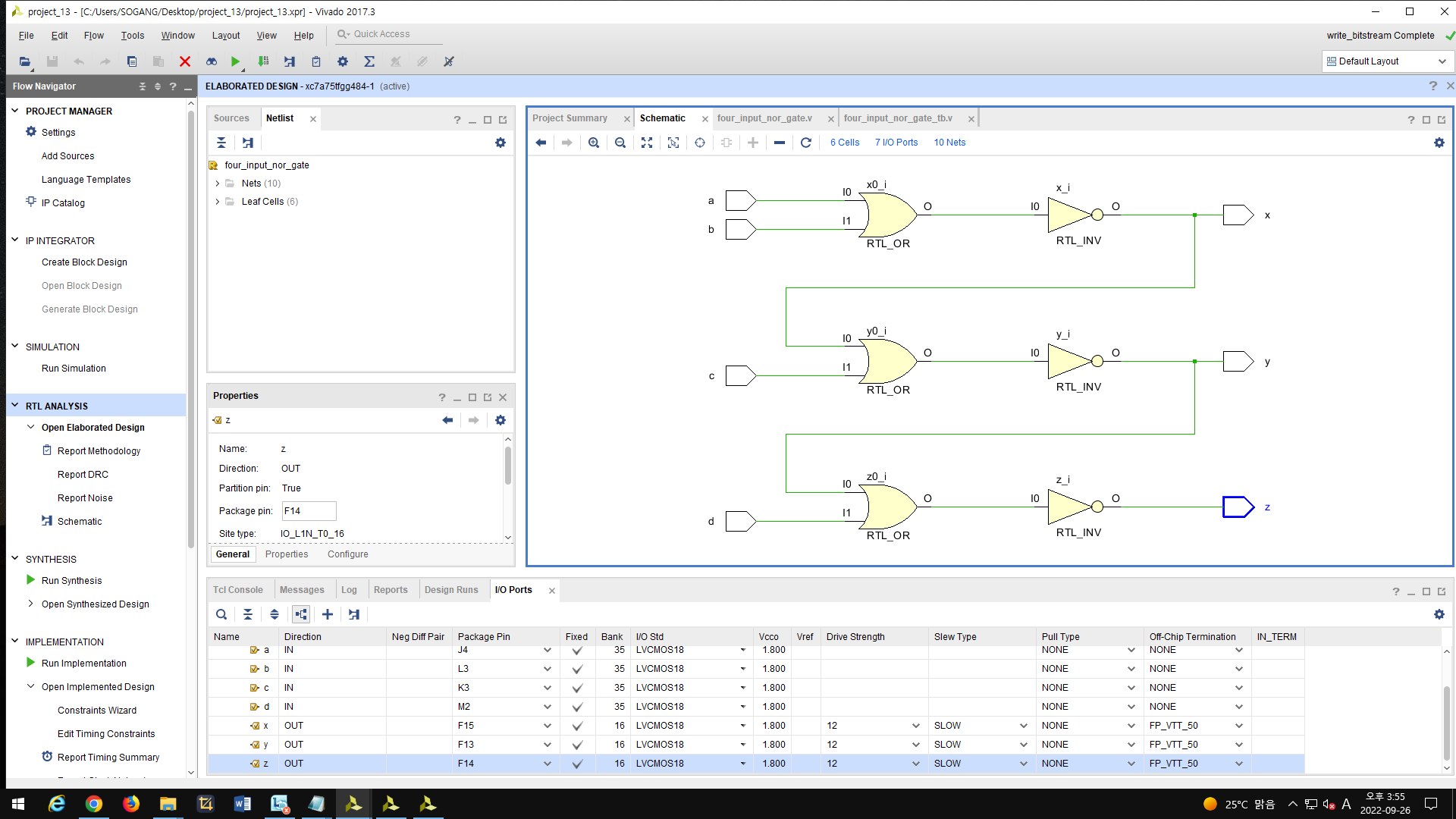
<그림7. 4-input NOR gate simulation>

시뮬레이션 소스 파일을 실행한 결과 다음과 같은 시뮬레이션 결과가 나왔다. NOR는 모든 입력이 0인 경우 1을, 나머지 경우에는 0을 출력한다. 따라서 xx는 aa와 bb가 모두 0인 경우, yy는 xx와 cc가 모두 0인 경우, zz는 yy와 dd가 모두 0인 경우 1을 출력하고, 나머지 경우에는 모두 0을 출력한다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input a | Input b | Input c | Input d | Output x | Output y | Output z |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

<table2. 4-input NOR gate truth table>

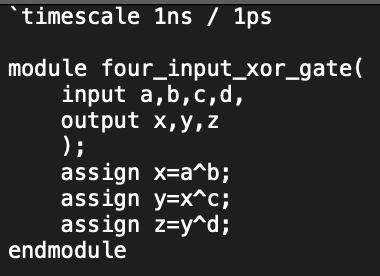
NOR gate의 시뮬레이션 결과를 진리표로 정리하였다.



<그림8. 4-input NOR gate schematic>

위 그림은 NOR gate를 schematic형태로 구현한 모습이다. OR gate와 NOT gate를 지나며 NOR게이트가 됨을 알 수 있다.

**4.**



<그림9. 4-input XOR gate design source code>

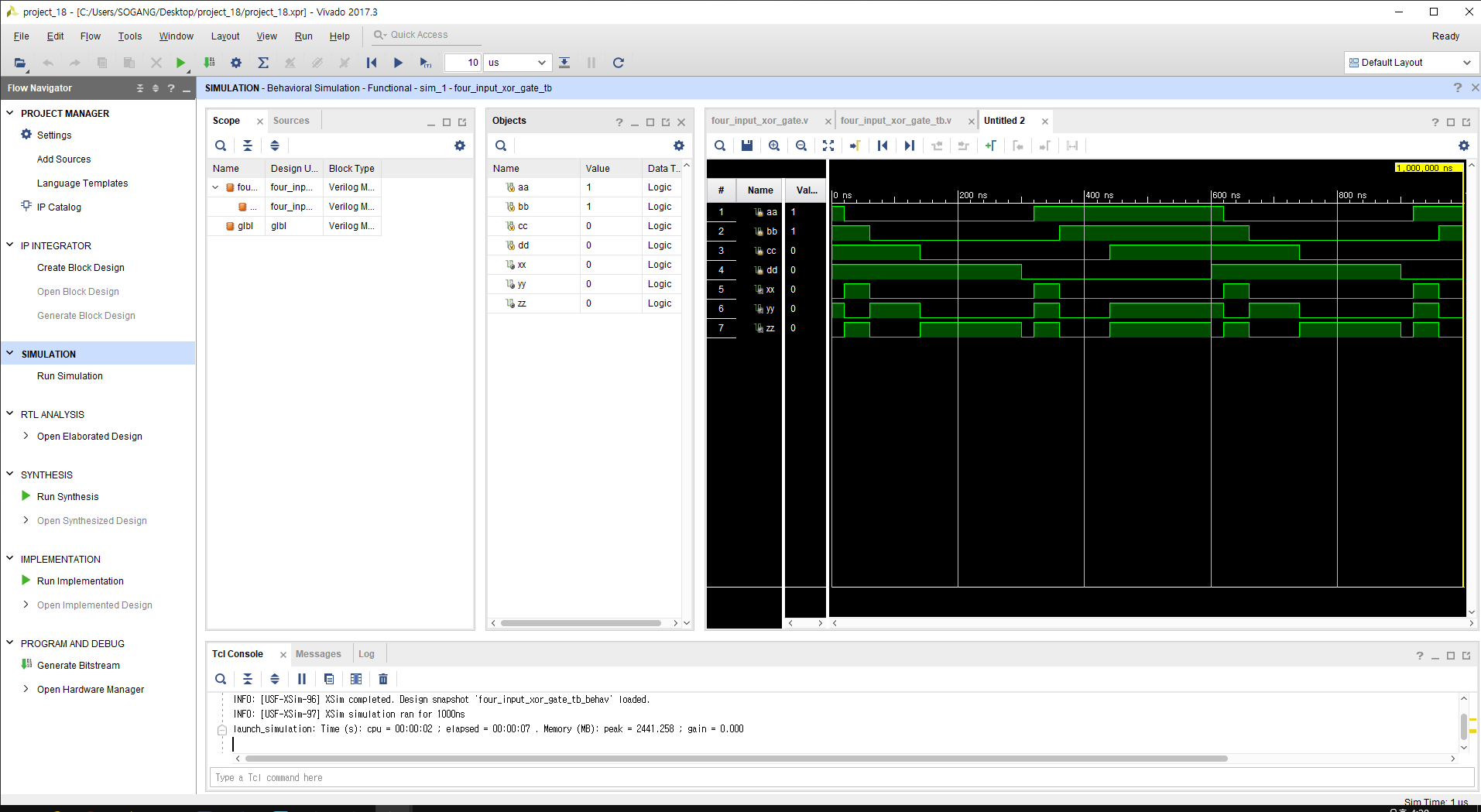
위 그림은 4개의 입력과 3개의 출력을 가지는 XOR gate의 디자인 소스 코드이다. 변수 x는 입력 a와 b의 XOR 연산을 한 결과인 a ^ b를 저장하고 있으며, 변수 y는 앞서 연산 결과인 x와 c의 XOR 연산 결과 x ^ c를, 변수 z는 앞선 연산 결과인 y와 d의 XOR 연산 결과 y ^ d를 저장하고 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림10. 4-input XOR gate simulation source code>

위 그림은 디자인 소스 코드에서 구현한 4\_input\_XOR\_gate를 시뮬레이션으로 구현하기 위한 시뮬레이션 코드이다. 입력값에 값을 할당하고, 값이 변경되는 주기와, 전체 시간을 설정하였다.



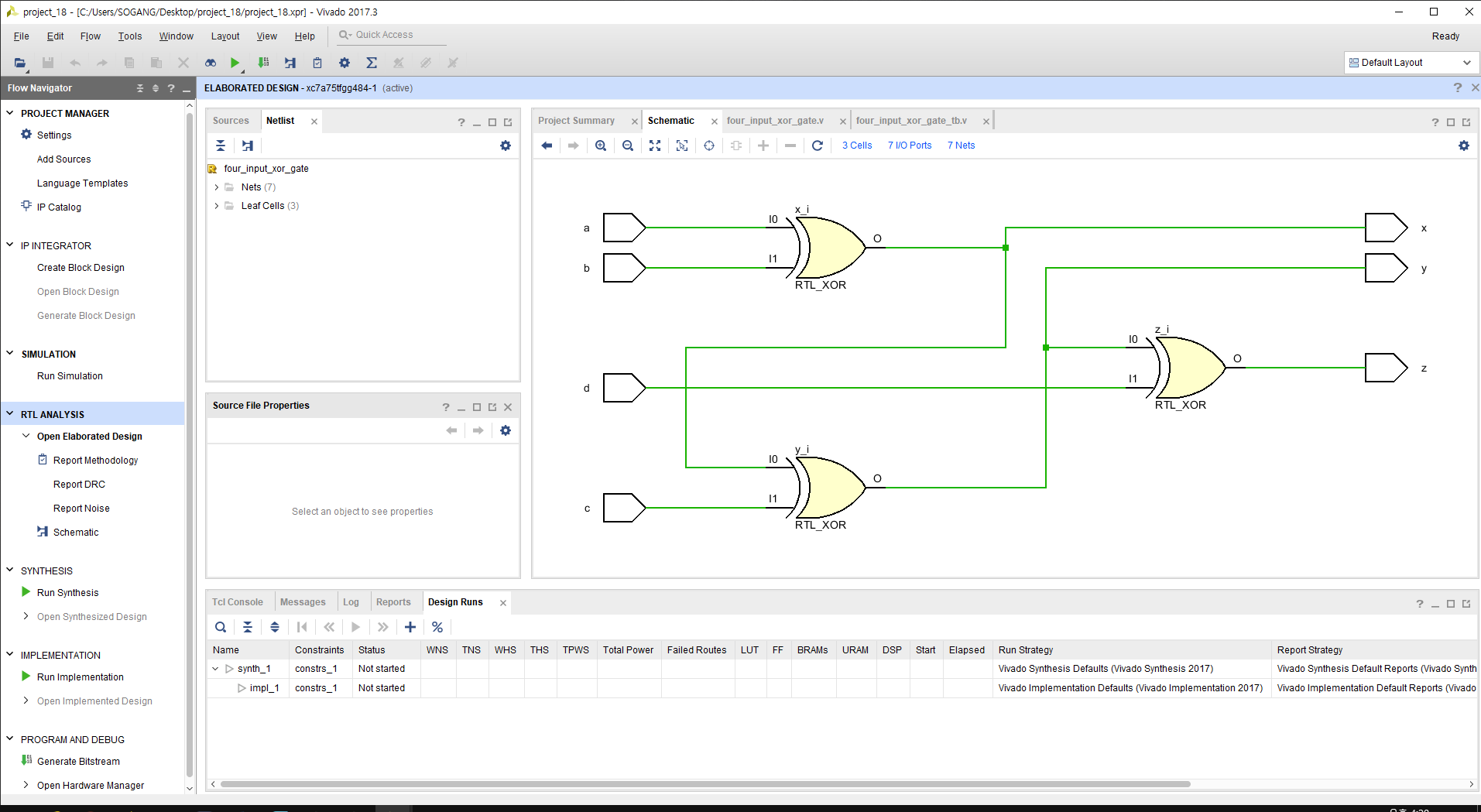
<그림11. 4-input XOR simulation>

시뮬레이션 소스 파일을 실행한 결과 다음과 같은 시뮬레이션 결과가 나왔다. XOR는 입력 값의 개수가 홀수일 때, 1을, 짝수일 때 0을 출력한다. 따라서 xx는 aa와 bb가 같은 값을 가지는 경우 0을, 다른 값을 가지는 경우 1을, yy는 xx와 cc가 같은 값을 가지는 경우 0을, 다른 값을 가지는 경우 1을, zz는 yy와 dd가 같은 값을 가지는 경우 0을 다른 값을 가지는 경우 1을 출력한다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input a | Input b | Input c | Input d | Output x | Output y | Output z |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

<table3. 4-input XOR gate truth table>

XOR gate의 시뮬레이션 결과를 진리표로 정리하였다.



<그림12. 4-input XOR schematic>

위 그림은 XOR gate를 schematic형태로 구현한 모습이다. XOR gate하나로 표현이 가능하다.

**5.**

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림13. 4-input AOI gate design source code>

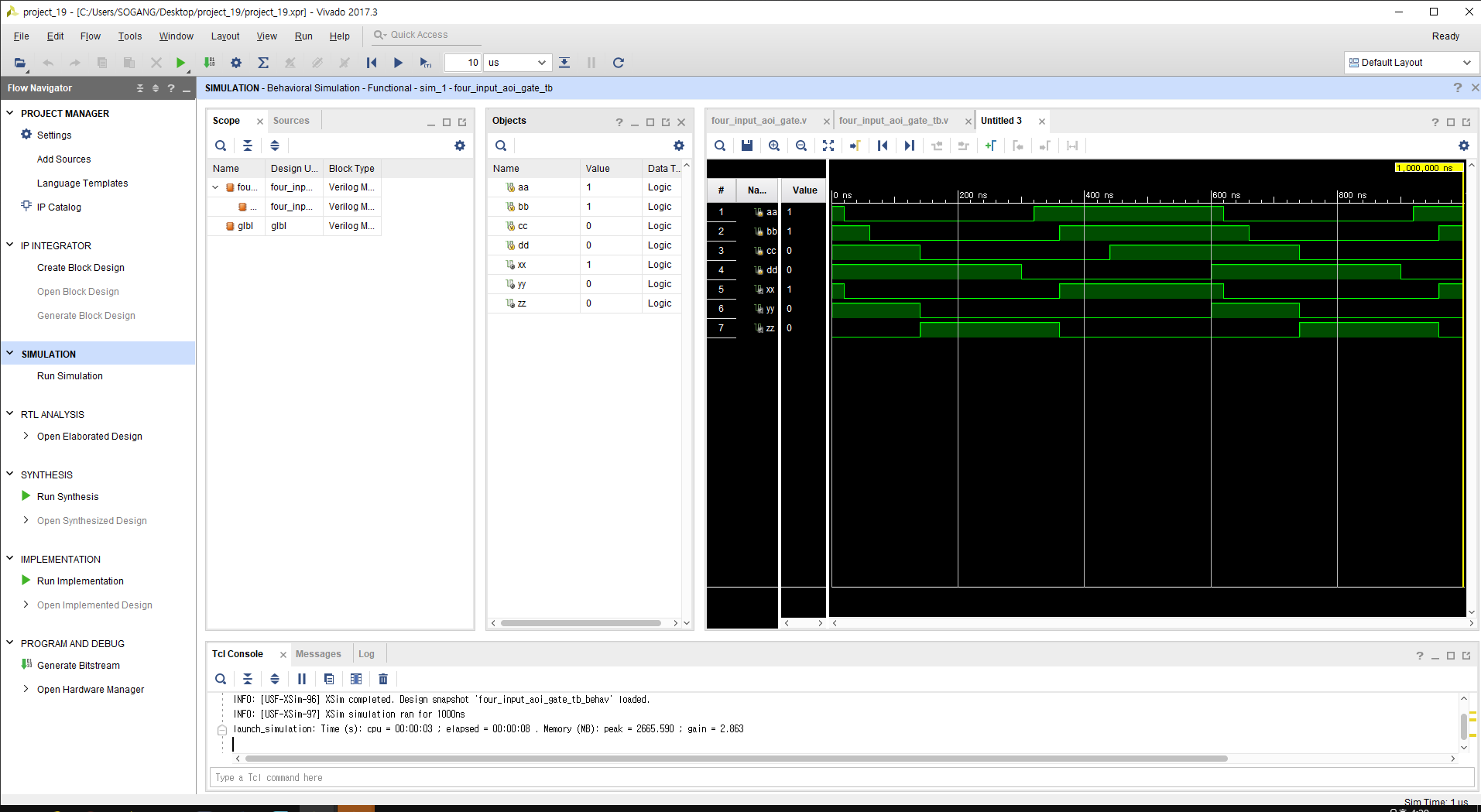
위 그림은 4개의 입력과 3개의 출력을 가지는 AOI gate의 디자인 소스 코드이다. 변수 x는 입력 a와 b의 AND연산을 한 결과인 a & b를 저장하고 있으며, 변수 y는 c와 d의 AND 연산 결과 c & d를, 변수 z는 앞선 연산 결과인 x와 y의 NOR 연산 결과 ~(x | y)를 저장하고 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림14. 4-input AOI gate simulation source code>

위 그림은 디자인 소스 코드에서 구현한 4\_input\_AOI\_gate를 시뮬레이션으로 구현하기 위한 시뮬레이션 코드이다. 입력값에 값을 할당하고, 값이 변경되는 주기와, 전체 시간을 설정하였다.



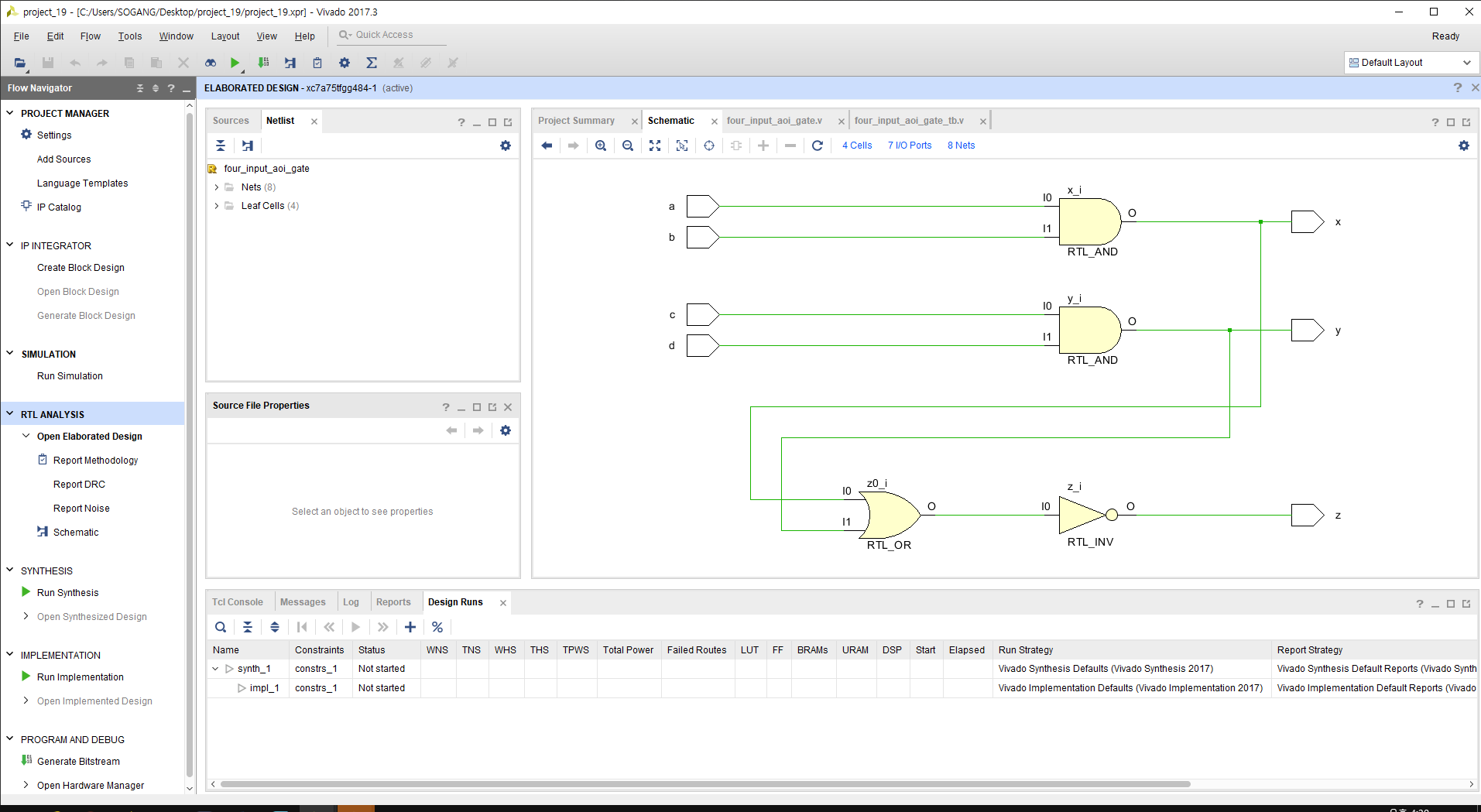
<그림15. 4-input AOI gate simulation>

시뮬레이션 소스 파일을 실행한 결과 다음과 같은 시뮬레이션 결과가 나왔다. AOI는 두개의 AND gate와 하나의 XOR gate를 사용한다. 따라서 xx는 aa와 bb의 AND 연산으로 aa와 bb가 모두 1을 가지는 경우 1을, 그렇지 않은 경우 0을 출력한다. yy는 cc와 dd의 AND 연산으로 cc와 bb가 모두 1을 가지는 경우 1을, 그렇지 않은 경우 0을 출력한다. zz는 xx와 yy의 XOR 연산으로 xx와 yy가 모두 0인 경우 1을, 그렇지 않은 경우 0을 출력한다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input a | Input b | Input c | Input d | Output x | Output y | Output z |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |

<table4. 4-input AOI gate truth table>

AOI gate의 시뮬레이션 결과를 진리표로 정리하였다.



<그림16. 4-input AOI gate schematic>

위 그림은 AOI gate를 schematic형태로 구현한 모습이다. 2개의 AND gate와 하나의 XOR gate로 표현되며, 4개의 입력을 2개씩 AND 연산 후, 두 AND 연산의 결과를 OR gate와 NOT gate를 지나게 해 XOR 결과를 얻는다.

**6.**

4개의 입력을 가지는 NAND, NOR, XOR, AOI gate에 대하여 디자인 소스 코드, 시뮬레이션 소스 코드를 구현하고, simulation을 생성하였다. 이후 SYNTHESIS와 IMPLEMENTATION, BITSTREAM 생성을 통해 FPGA에서 동작을 수행하였다. NAND gate는 AND의 부정으로, 모든 입력이 1일 때 0을, 나머지 경우에 대해서는 1을 출력한다. NOR gate는 OR의 부정으로 모든 입력이 0일 때 1을, 나머지 경우에는 0을 출력한다. XOR gate는 입력된 1의 개수가 홀수일 때 1을, 짝수일 때 0을 출력한다. AOI gate는 두 AND gate의 값을 XOR 연산하여 값을 출력하였다.

**7.** 추가 이론 조사 및 작성

위 실험은 XOR을 a ^ b와 같이 표현하였다. 하지만 AND, OR, NOT gate를 활용하여 (a & ~b) | (~a & b)로 표현할 수 있다. 이는 진리표를 이용하여 두 식이 같음을 알 수 있다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | a^b | ~a | ~b | a&~b | ~a&b | (a&~b)|(~a&b) |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

뿐만 아니라 (a & ~b) | (~a & b)는 (a | b) & (~a | ~b)와 같음을 알 수 있고, 이로 인해 a와 b의 XOR 연산 a ^ b = (a | b) & (~a | ~b)로 표현할 수 있다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | a^b | ~a | ~b | a|b | ~a|~b | (a|b)&(~a|~b) |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |