5주차 결과보고서

전공: 물리학과 학년: 3학년 학번: 20191286 이름: 김나현

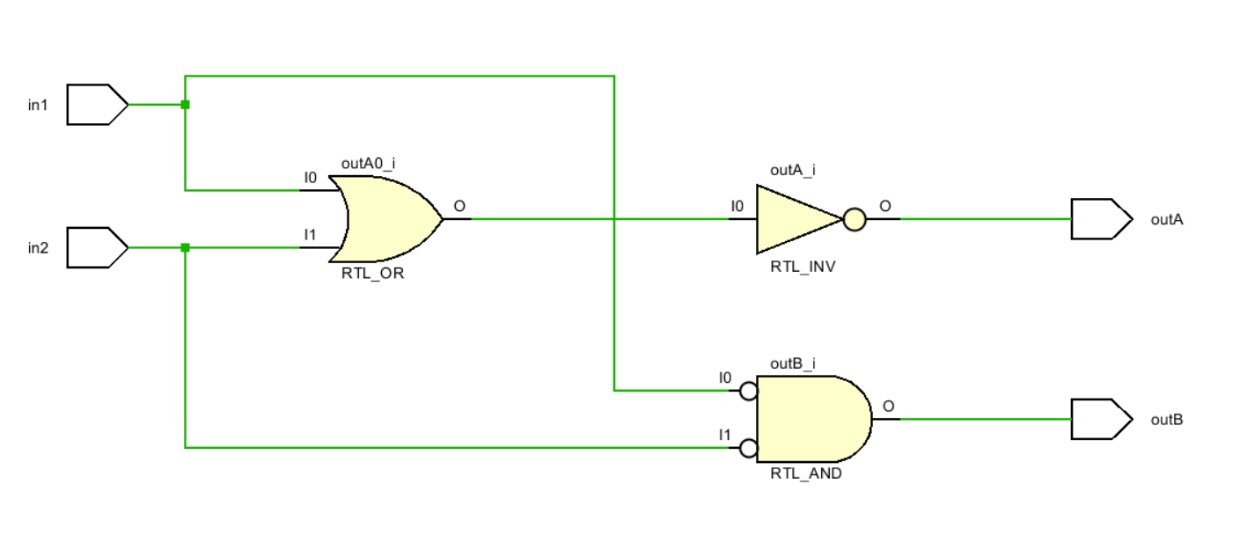
1. 실험 목적

De Morgan의 정리를 확인하기 위해 Verilog를 사용하여 (A+B)’=A’B’, (AB)’=A’+B’라는 사실을 직접 확인해보고, De Morgan 정리를 확인할 수 있는 더욱 복잡한 Boolean 함수의 동작을 확인합니다. 또한, 1 bit 비교기의 Verilog 코드를 직접 짜보고 확인해 봄으로써 진리표와 SOP를 이용해서 논리 함수를 만드는 방법을 알아봅니다.

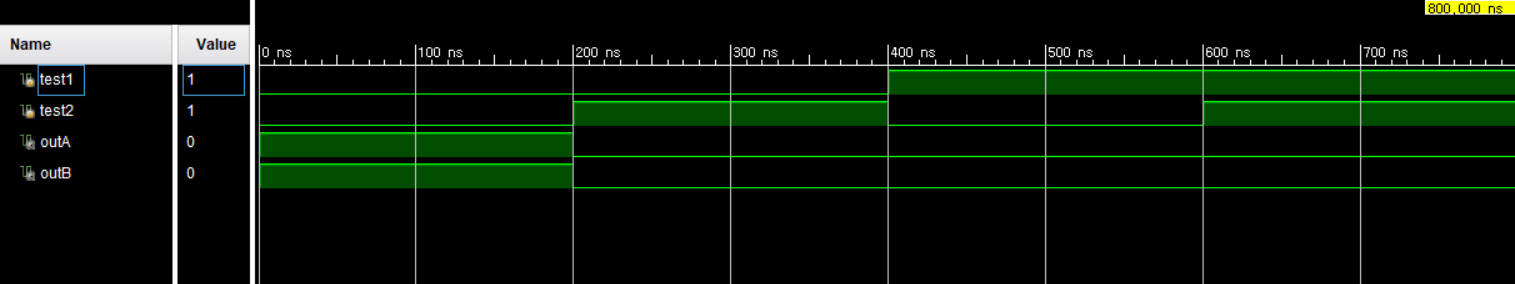
1. De Morgan의 제1,2법칙의 simulation 결과 및 과정에 대해서 설명하시오(NAND, NOR과 비교 포함).

텍스트이(가) 표시된 사진

자동 생성된 설명

<Figure 1> 드 모르간 제1법칙 Verilog 코드

<Figure 2> 드 모르간 제1법칙 schematic diagram



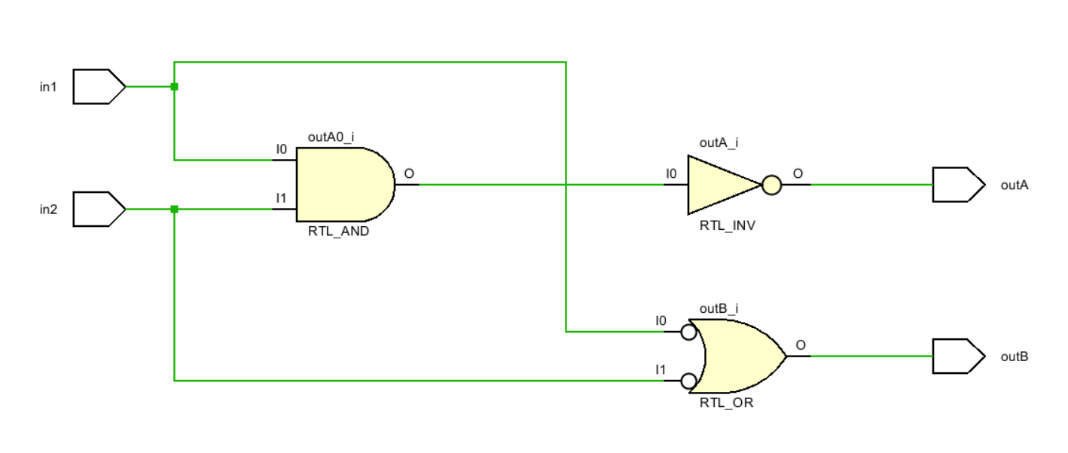
<Figure 3> 드 모르간 제1법칙 simulation 결과

드 모르간 법칙 중 제1법칙을 확인하기 위하여 위 <Figure 1>과 같이 Verilog 코딩을 하였습니다. 드 모르간 제1법칙은 두 개의 입력을 논리합한 뒤, 보수를 취한 것과 두 개의 입력의 보수를 각각 얻어 두 값을 논리곱한 것이 같다는 것입니다. <Figure 1>에서 두 개의 입력, in1, in2에 대해 outA=(inA+inB)’와 outB=(inA’)(inB’)을 assign하여 schematic을 확인해보고, testbench를 만들어 simulation 결과를 얻어보았습니다. <Figure 2>를 보면, outA는 두 개의 입력 in1와 in2를 논리합한 뒤, 그 값을 NOT 게이트에 통과시킨 결과이고 outB는 두 입력을 각각 NOT 게이트에 통과시킨 후, 논리곱한 결과임을 알 수 있습니다. <Figure 3>은 <Figure 1>의 Verilog 코드를 testbench를 이용해 simulation한 결과로, outA와 outB가 모든 입력 경우에 대해 똑같은 출력을 보인다는 사실로부터 드 모르간 제1법칙이 실제로 적용된 모습을 확인할 수 있습니다. 또한 <Figure 3>을 진리표로 작성하면 NOR 게이트의 진리표와 같다는 것을 알 수 있습니다. 따라서, NOR 연산은 (A+B)’이지만 A’B’로도 구할 수 있다는 결론을 내릴 수 있습니다.

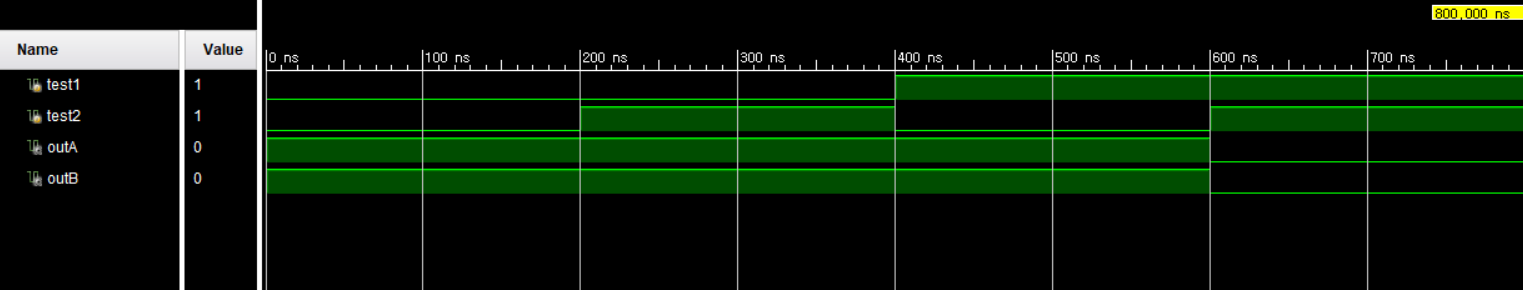
텍스트이(가) 표시된 사진

자동 생성된 설명

<Figure 4> 드 모르간 제1법칙 Verilog 코드



<Figure 5> 드 모르간 제2법칙 schematic diagram



<Figure 6> 드 모르간 제2법칙 simulation 결과

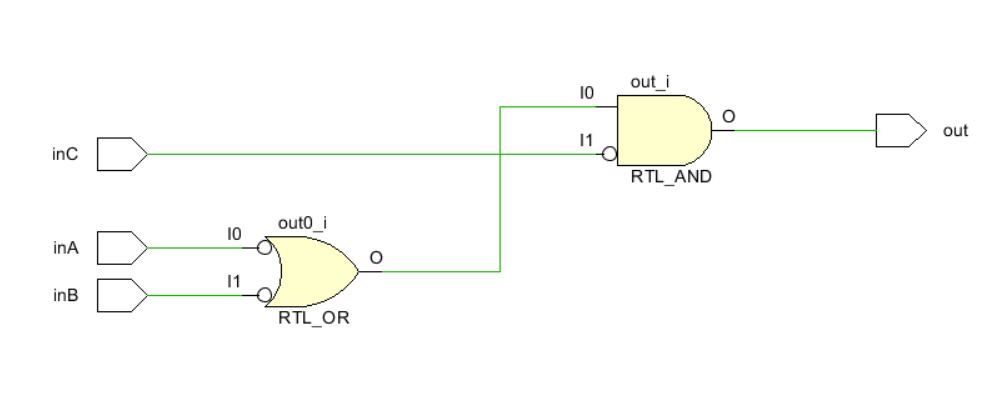
마찬가지로, 드 모르간 법칙 중 제2법칙을 확인하기 위하여 위 <Figure 4>과 같이 Verilog 코딩을 하였습니다. 드 모르간 제2법칙은 두 개의 입력을 논리곱한 뒤, 보수를 취한 것과 두 개의 입력의 보수를 각각 얻어 두 값을 논리합한 것이 같다는 것입니다. <Figure 4>에서 두 개의 입력, in1, in2에 대해 outA=((inA)(inB))’와 outB=(inA’)+(inB’)을 assign하여 schematic을 확인해보고, testbench를 만들어 simulation 결과를 얻어보았습니다. <Figure 5>를 보면, outA는 두 개의 입력 in1와 in2를 논리곱한 뒤, 그 값을 NOT 게이트에 통과시킨 결과이고 outB는 두 입력을 각각 NOT 게이트에 통과시킨 후, 논리합한 결과임을 알 수 있습니다. <Figure 6>은 <Figure 4>의 Verilog 코드를 testbench를 이용해 simulation한 결과로, outA와 outB가 모든 입력 경우에 대해 똑같은 출력을 보인다는 사실로부터 드 모르간 제2법칙이 실제로 적용된 모습을 확인할 수 있습니다. 또한 <Figure 6>을 진리표로 작성하면 NAND 게이트의 진리표와 같다는 것을 알 수 있습니다. 따라서, NAND 연산은 (AB)’이지만 A’+B’로 바꿔서 연산할 수 있다는 결론을 내릴 수 있습니다.

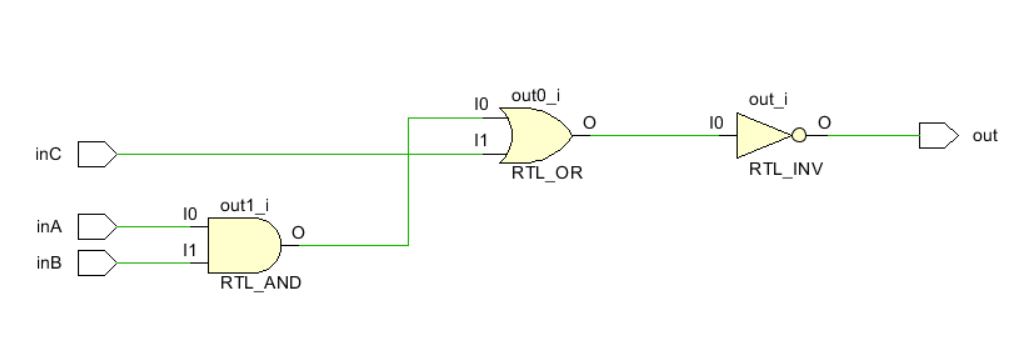
<Table 1> 드 모르간의 정리 진리표

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | A+B | AB | A’ | B’ | (A+B)’ | A’B’ | (AB)’ | A’+B’ |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

위의 <Table 1>은 드 모르간의 법칙을 정리한 진리표로, AND 게이트의 보수인 NAND 게이트는 (AB)’ 대신 A’+B’로 연산할 수 있고, OR 게이트의 보수인 NOR 게이트는 (A+B)’ 대신 A’B’로 연산할 수 있다는 것을 알 수 있습니다.

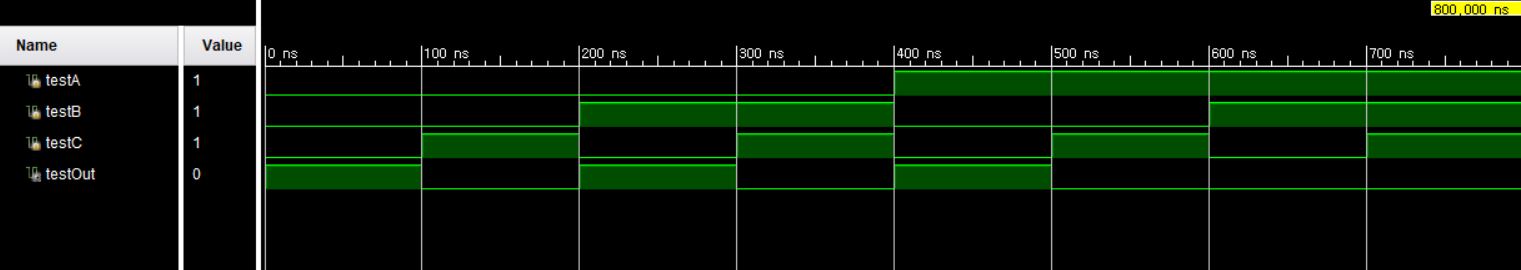
1. (A’+B’)·C’=(A·B+C)’의 simulation 결과 및 과정에 대해서 설명하시오(+ 및 · 위치를 바꾼 모양도 수행)

AB를 D라고 했을 때, (AB+C)’는 (D+C)’가 되고, 이는 앞서 말한 드 모르간의 제1법칙에 의해 D’C’가 됩니다. 이때, 다시 D’은 (AB)’이므로 드 모르간의 제2법칙에 의해 A’+B’이므로 D’C’는 (A’+B’)C’이 됩니다. 따라서, 두 boolean function (AB+C)’와 (A’+B’)C’가 같다는 사실을 실험적으로 알아보기 위해 Verilog 코딩할 수 있습니다.

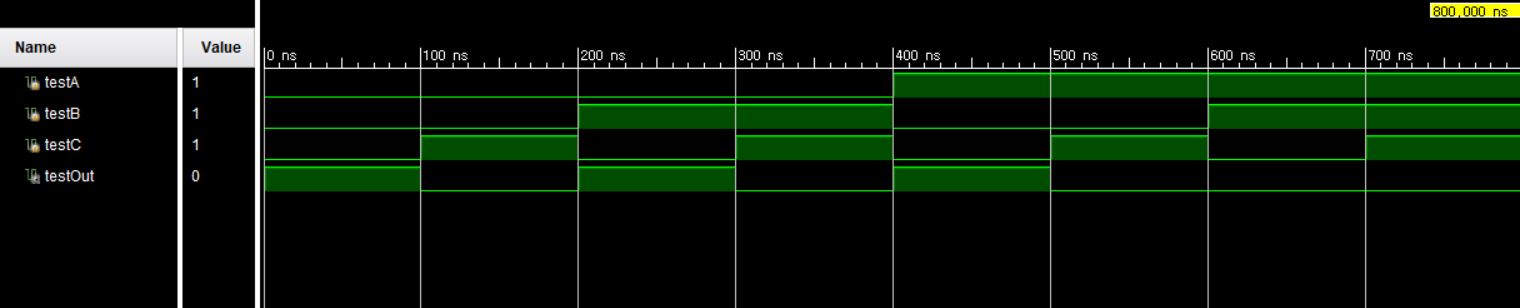
<Figure 7> (A’+B’)·C’의 schematic diagram

<Figure 8> (A·B+C)’의 schematic diagram

<Figure 7>과 <Figure 8>은 각각 (A’+B’)·C’와 (A·B+C)’ Boolean function의 schematic 모습입니다. 두 개의 schematic은 완전히 달라 보이지만 이 두 함수의 simulation 결과를 확인해보면 두 개의 함수가 모든 입력 경우에 대해 똑같은 출력을 보인다는 사실을 확인할 수 있습니다.



<Figure 9> (A’+B’)·C’의 simulation 결과



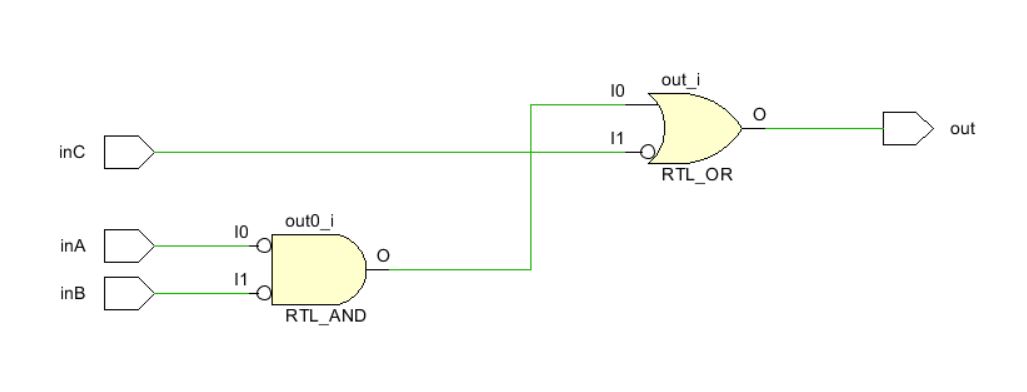
<Figure 10> (A·B+C)’의 simulation 결과

위의 두 simulation 결과를 보면 완벽히 동일한 출력을 갖는다는 것을 알 수 있습니다. 이를 진리표로 작성해보면 다음 <Table 2>와 같습니다.

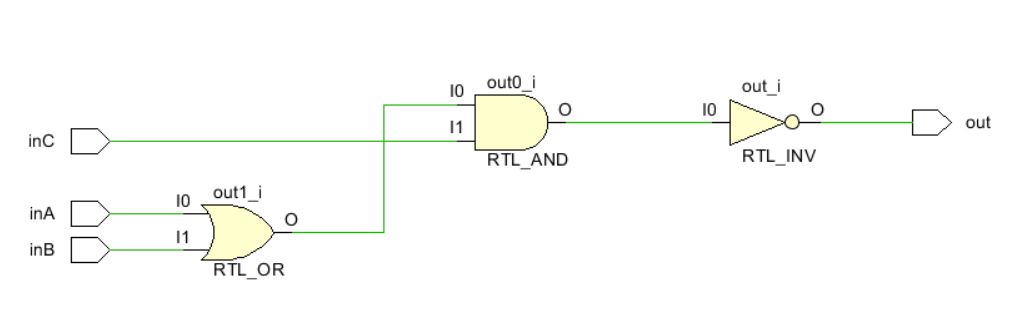
<Table 2> (A·B+C)’ 또는 (A’+B’)·C’ Boolean function의 진리표

|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | | | Output |
| A | B | C | D |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

(A’+B’)·C’의 출력이 1이 되려면 C가 무조건 0이고, A와 B의 논리곱이 0이어야 하므로 A와 B 둘 중 하나 이상이 0이어야 합니다. 따라서, 위의 <Table 2>처럼 (A, B, C)가 (0, 0, 0)일 때나, (0, 1, 0), (1, 0, 0)일 때만 출력이 1이 됩니다. 마찬가지로 (A·B+C)’의 출력이 1이 되려면 AB+C가 0이어야 하므로 C는 무조건 0이고, A와 B의 논리곱이 0이어야 하므로 (A’+B’)·C’의 출력이 1이 되기 위한 조건과 동일하므로 같은 출력을 보인다는 사실을 알 수 있습니다.

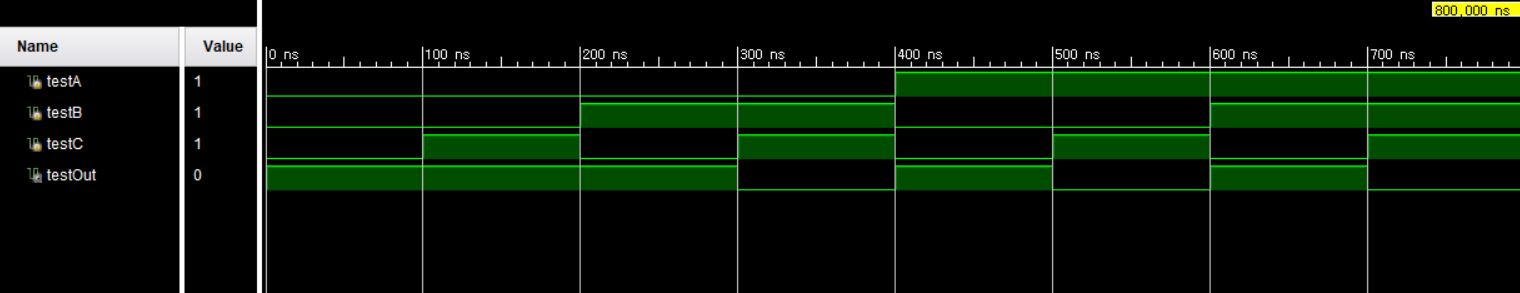
이제 A’B’+C’=((A+B)C)’를 확인하는 실험에 대해 설명해보겠습니다. A+B를 D라고 했을 때, (DC)’는 이는 드 모르간의 제2법칙에 의해 D’+C’가 됩니다. 이때, 다시 D’은 (A+B)’이므로 드 모르간의 제1법칙에 의해 A’B’이므로 D’+C’는 A’B’+C’이 됩니다. 이론적으로 같다고 확인한 두 boolean function, ((A+B)C)’와 A’B’+C’가 같다는 사실을 실험적으로 알아보기 위해 Verilog 코딩할 수 있습니다.

<Figure 11> A’B’+C’의 schematic diagram

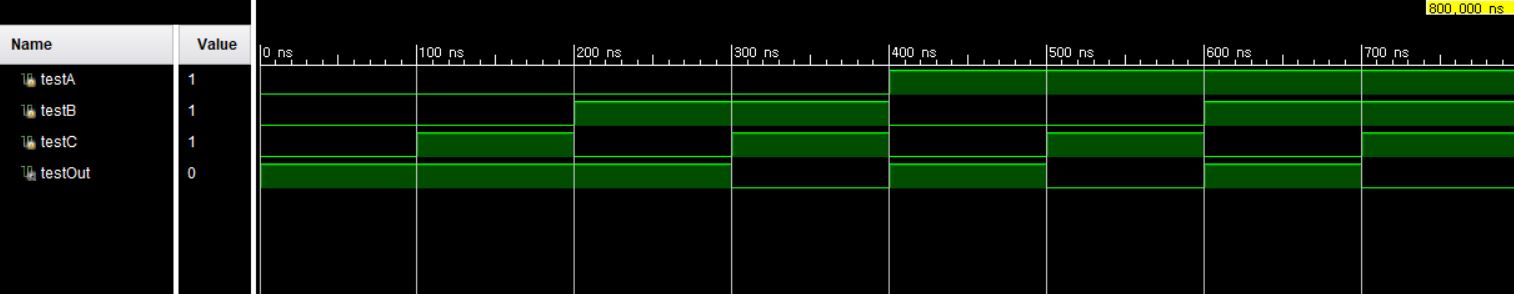


<Figure 12> ((A+B)C)’의 schematic diagram

<Figure 11>과 <Figure 12>은 각각 A’B’+C’와 ((A+B)C)’Boolean function의 schematic 모습입니다. 두 개의 schematic은 완전히 달라 보이지만 이 두 함수의 simulation 결과를 확인해보면 두 개의 함수가 모든 입력 경우에 대해 똑같은 출력을 보인다는 사실을 확인할 수 있습니다.



<Figure 13> A’B’+C’의 simulation 결과



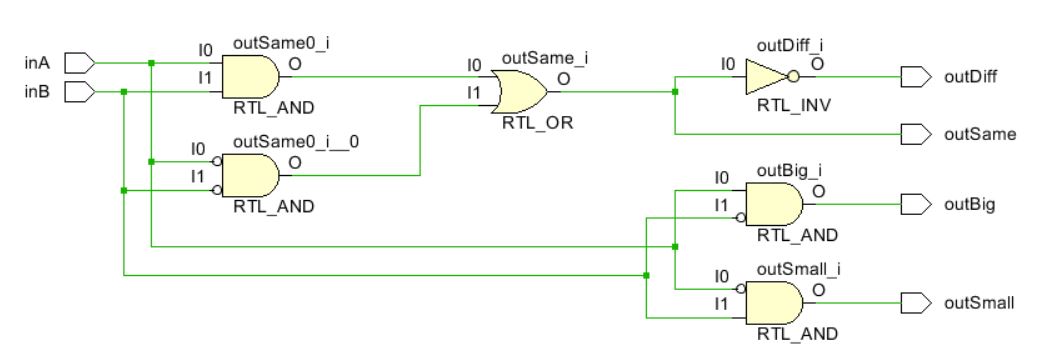
<Figure 14> ((A+B)C)’의 simulation 결과

위의 두 simulation 결과를 보면 완벽히 동일한 출력을 갖는다는 것을 알 수 있습니다. 이를 진리표로 작성해보면 다음 <Table 3>과 같습니다.

<Table 3> ((A+B)C)’또는 A’B’+C’ Boolean function의 진리표

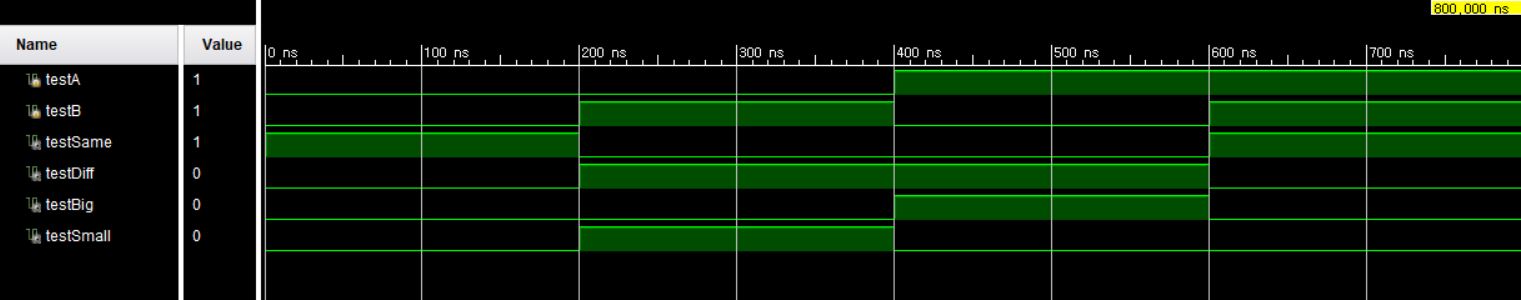
|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | | | Output |
| A | B | C | D |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

A’B’+C’의 출력이 1이 되는 경우는 A’B’가 1이거나 C’가 1이거나 둘 다 1인 경우로 구분됩니다. A’B’가 1이고, C’이 0인 경우는 A, B가 0이고 C가 1인 경우이고, A’B’가 0이고 C’이 1인 경우는 (A, B, C)가 (0, 1, 0), (1, 0, 0), (1, 1, 0)인 경우이고, 둘 다 1인 경우는 (0, 0, 0)인 경우입니다. 따라서, <Table 3>과 같은 진리표를 얻을 수 있습니다. ((A+B)C)’의 출력이 1이 되는 경우 또한 (A+B)C가 0인 경우이므로, (0, 0, 0), (0, 1, 0), (1, 0, 0), (1, 1, 0), (0, 0, 1)이므로 A’B’+C’의 출력이 1이 되는 경우와 동일하므로 같은 진리표를 얻는다는 사실을 알 수 있습니다.

1. 1 bit 비교기의 simulation 결과 및 과정에 대해서 설명하시오(2 input, 4 output)(진리표 작성).

<Figure 15>1 bit 비교기의 schematic diagram

위의 <Figure 15>는 1 bit 비교기의 schematic diagram의 모습입니다. 이때, outDiff는 두 개의 입력, inA와 inB가 서로 다르면 1을 출력하고, outSame은 두 개의 입력이 서로 같으면 1을 출력하고, outBig은 inA가 inB보다 크면 1을, outSmall은 inA가 inB보다 작으면 1을 출력하도록 verilog 코딩한 것입니다.



<Figure 16>1 bit 비교기의 simulation

위의 <Figure 16>는 <Figure 15>의 1 bit 비교기의 simulation 결과입니다. testSame을 보면 testA와 testB가 같을 때마다 1을 출력하고, testDiff를 보면 testA와 testB가 서로 다를 때마다 1을 출력하고, testBig은 testA가 1이고, testB가 0일 때 1을 출력하고, testSmall은 testA가 0이고 testB가 1일 때 1을 출력하는 것을 알 수 있습니다.

<Table 4>1 bit 비교기의 진리표

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | outSame | outDiff | outBig | outSmall |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

위의 <Table 4>은 1 bit 비교기의 진리표를 나타낸 것입니다. 이 진리표를 보고 verilog 코드를 짤 수 있는데 바로 예비 보고서에 작성한 SOP, 즉 sum of product를 이용하여 적는 것입니다. outSame은 두 입력 A와 B가 서로 같을 때이므로 AB+A’B’로 구할 수 있고, outDiff는 두 입력이 서로 다를 때이므로 AB’+A’B로 구할 수 있고, outBig은 AB’을, outSmall은 A’B를 이용하여 구할 수 있습니다. 따라서 이러한 Verilog 코드를 이용해 schematic을 확인하면 <Figure 15>과 같고, testbench를 이용하여 simulation 결과를 확인하면 <Figure 16>와 같게 됩니다.

1. 결과 검토 및 논의사항

본 실험을 통해 드 모르간의 제1, 2법칙을 실험적으로 확인해보고, 주어진 진리표를 보고 minterm(product)을 이용한 합(SOP) 또는 maxterm을 이용한 곱을 통해 논리 함수로 구현해보았습니다. 드 모르간의 법칙은 논리합의 보수는 입력의 보수를 논리곱을 하는 것과 논리곱의 보수는 입력의 보수를 논리합하는 것과 같다는 것이고, 간단한 Verilog 코드를 통해 (A+B)’=A’B’, (AB)’=A’+B’라는 것을 확인해볼 수 있었습니다. 또한 드 모르간의 법칙에 의해 논리 함수가 다양한 형태로 표현될 수 있다는 사실도 확인해볼 수 있었습니다. 마지막으로 1 bit 비교기를 직접 구현해봄으로써 1 bit 비교기의 역할, 즉 진리표를 미리 알고 있는 상태에서 논리 함수를 구현하여 Verilog 코딩하는 실습을 해보았습니다.

1. 추가 이론 조사 및 작성

1 bit 비교기의 진리표, <Table 4>를 보고 카르노 맵을 이용하여 논리 함수를 찾아보겠습니다.

<Table 5> 1 bit 비교기의 outSame

|  |  |  |
| --- | --- | --- |
| B/A | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

위의 <Table 5>를 보고 prime implicant을 찾으면 outSame 함수는 A’B’와 AB의 minterm의 합으로 표현될 수 있다는 것을 알 수 있습니다.

<Table 6> 1 bit 비교기의 outDiff

|  |  |  |
| --- | --- | --- |
| B/A | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1 | 0 |

마찬가지로 위의 <Table 6>을 보고 prime implicant을 찾으면 outDiff 함수는 A’B와 AB’의 minterm의 합으로 표현될 수 있다는 것을 알 수 있습니다. 이는 XOR 연산과 동일합니다.

<Table 7> 1 bit 비교기의 outBig

|  |  |  |
| --- | --- | --- |
| B/A | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 0 | 0 |

A가 B보다 클 때 1을 출력하는 outBig은 AB’일 때만 1을 출력하므로 minterm 하나로, outBig=AB’로 표현이 가능합니다.

<Table 7> 1 bit 비교기의 outSmall

|  |  |  |
| --- | --- | --- |
| B/A | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 0 |

A가 B보다 작을 때 1을 출력하는 outSmall은 A’B일 때만 1을 출력하므로 minterm 하나로, outSmall=A’B로 표현이 가능합니다.

이처럼 두 개의 입력보다 더 많은 입력에 대한 진리표가 주어졌을 때도 카르노 맵의 방법을 이용하여 논리 함수 식을 찾을 수 있습니다.