전공: 컴퓨터공학 학년: 2학년 학번: 20161603 이름: 신민준

1. 실험 목적
2. De-Morgan의 제 1, 2 법칙의 simulation 결과 및 과정에 대해서 설명하시오. (NAND, NOR 과 비교 포함)

* De Morgan 제 1법칙

De Morgan 제 1법칙의 우변과 좌변을 구현한 Verilog code는 다음과 같다.

|  |
| --- |
| `timescale 1ns / 1ps  module dm1A(      input a, b,      output out      );      assign out = ~(a|b);  endmodule  module dm1B(      input a, b,      output out      );      assign out = (~a)&(~b);  endmodule |

이 코드의 RTL Schematic을 각각 보이면 다음과 같다.

좌변 의 Schematic은 NOR의 Schematic과 동일한 것을 확인할 수 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

또한, 우변의 Schematic을 통해 두 input이 invert된 AND gate는 NOR gate와 동일하단 것을 확인할 수 있다.

구현한 Design들을 Simulation할 때 사용한 simulation source는 다음과 같다.

|  |
| --- |
| `timescale 1ns / 1ps  module sim;      reg [1:0] in;      wire outA, outB;      dm1A testA(in[0], in[1], outA);      dm1B testB(in[0], in[1], outB);        initial in = 2'b00;      always in = #200 in + 1;      initial begin          #20000          $finish;      end  endmodule |

위 simulation source를 사용해서 나온 결과는 다음과 같다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

* De Morgan 제 2법칙

De Morgan 제 2법칙의 우변과 좌변을 구현한 Verilog code는 다음과 같다.

|  |
| --- |
| `timescale 1ns / 1ps  module dm2A(      input a, b,      output out      );      assign out = ~(a&b);  endmodule  module dm2B(      input a, b,      output out      );      assign out = (~a)|(~b);  endmodule |

이 코드의 RTL Schematic을 각각 보이면 다음과 같다.

좌변 의 Schematic은 NAND의 Schematic과 동일한 것을 확인할 수 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

또한, 우변의 Schematic을 통해 두 input이 invert된 OR gate는 NAND gate와 동일하단 것을 확인할 수 있다.

구현한 Design들을 Simulation할 때 사용한 simulation source는 다음과 같다.

|  |
| --- |
| `timescale 1ns / 1ps  module sim;      reg [1:0] in;      wire outA, outB;      dm2A testA(in[0], in[1], outA);      dm2B testB(in[0], in[1], outB);        initial in = 2'b00;      always in = #200 in + 1;      initial begin          #20000          $finish;      end  endmodule |

위 simulation source를 사용해서 나온 결과는 다음과 같다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

De Morgan의 1, 2법칙의 Simulation의 결과를 정리해 진리표를 구했다.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **A+B** | **A•B** | **A’** | **B’** | **(A+B)’** | **A’•B’** | **(A•B)’** | **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 |

1. 의 simulation 결과 및 과정에 대해서 설명하시오. ( 및 위치 바꾼 모양도 수행)

위 binary function의 좌변과 우변을 구현한 Verilog code는 다음과 같다.

|  |
| --- |
| `timescale 1ns / 1ps  module boolf1A(      input a, b, c,      output out      );      assign out = (~a|~b)&~c;  endmodule  module boolf1B(      input a, b, c,      output out      );      assign out = ~((a&b)|c);  endmodule |

이 코드의 RTL Schematic을 각각 보이면 다음과 같다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

구현한 Design들을 Simulation할 때 사용한 simulation source는 다음과 같다.

|  |
| --- |
| module boolsim;      reg [2:0] in;      wire outA, outB;      boolf1A testA(in[0], in[1], in[2], outA);      boolf1B testB(in[0], in[1], in[2], outB);        initial in = 3'b000;      always in = #200 in + 1;      initial begin          #20000          $finish;      end  endmodule |

위 simulation source를 사용해서 나온 결과는 다음과 같다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

Simulation의 결과를 진리표로 정리하면 다음과 같은 결과가 나온다.

|  |  |  |  |
| --- | --- | --- | --- |
| **IN A** | **IN B** | **IN C** | **OUT 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 |

위 binary function의 좌변과 우변을 구현한 Verilog code는 다음과 같다.

|  |
| --- |
| `timescale 1ns / 1ps  module boolf2A(      input a, b, c,      output out      );      assign out = (~a&~b)|~c;  endmodule  module boolf2B(      input a, b, c,      output out      );      assign out = ~((a|b)&c);  endmodule |

이 코드의 RTL Schematic을 각각 보이면 다음과 같다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

구현한 Design들을 Simulation할 때 사용한 simulation source는 다음과 같다.

|  |
| --- |
| module boolsim;      reg [2:0] in;      wire outA, outB;      boolf2A testA(in[0], in[1], in[2], outA);      boolf2B testB(in[0], in[1], in[2], outB);        initial in = 3'b000;      always in = #200 in + 1;      initial begin          #20000          $finish;      end  endmodule |

위 simulation source를 사용해서 나온 결과는 다음과 같다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

Simulation의 결과를 진리표로 정리하면 다음과 같은 결과가 나온다.

|  |  |  |  |
| --- | --- | --- | --- |
| **IN A** | **IN B** | **IN C** | **OUT 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 |

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

1Bit 비교기의 진리표를 작성하면 다음과 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **A=B** | **A≠B** | **A>B** | **A<B** |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

이 진리표를 사용해 각각의 Karnaugh Map을 작성하면 다음과 같다.

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

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

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

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

위 Karnaugh Map에서 구한 함수를 Verilog Code로 다음처럼 구현했다.

|  |
| --- |
| `timescale 1ns / 1ps  module cmp(      input a, b,      output eq, neq, lg, le      );      assign eq = (~a&~b)|(a&b);      assign neq = (~a&b)|(a&~b);      assign lg = a&~b;      assign le = ~a&b;  endmodule |

이를 Schematic으로 보이면 다음과 같은 결과를 얻는다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

이 디자인을 simulation하기 위해 작성한 simulation code는 다음과 같다.

|  |
| --- |
| `timescale 1ns / 1ps  module cmpsim;      reg [1:0] in;      wire eq, neq, lg, le;      cmp testA(in[0], in[1], eq, neq, lg, le);        initial in = 2'b00;      always in = #200 in + 1;      initial begin          #20000          $finish;      end  endmodule |

Simulation 수행 후, 결과는 다음과 같이 보일 수 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

1. 결과 검토 및 논의사항

De Morgan의 1, 2법칙을 알아보기 위해 했던 실험에서, 두 실험 모두 양 변의 결과값에 차이가 전혀 없는 모습을 볼 수 있었고, 이를 통해 De Morgan의 1, 2법칙을 확인할 수 있었다.

더 나아가, Boolean Function에서의 양변이 simulation 결과 동일한 결과를 보인다는 것을 확인하며, De Morgan 법칙을 재확인 할 수 있었다.

1-bit Comparator의 경우, 각 함수가 의도된 대로 작동함을 볼 수 있었고, 이로부터 그린 Karnaugh Map이 정확하게 구현되었음을 확인했다.

각 Input을 위해 in[1:0] 배열을 선언해 사용했는데, 이 과정에서 입력 순서가 반대로 되어있어 진리표를 작성하는 과정이 살짝 불편했다. 인자로 넣는 과정에서 순서를 반대로 넣어주면 해결될 것이다.

1. 추가 이론 조사 및 작성

* De Morgan의 법칙을 사용해 복잡한 Boolean function을 invert할 때엔 다음과 같은 순서대로 하면 아무리 복잡한 함수도 쉽게 처리가 가능하다. 1. 먼저 Boolean function의 모든 literal들을 complement시킨다. 2. ‘’ 기호는 ‘’ 기호로, ‘’ 기호는 ‘’ 기호로 바꾼다.