**컴퓨터공학 설계 및 실험Ⅱ**

5주차 결과보고서

서강대학교 공학부 컴퓨터공학 전공

20171646 박태윤

**1. 실험 목적**

De-Morgan의 제1, 2 법칙에 관한 내용을 살펴보고 이를 Verilog로 구현한 뒤 Simulation을 통해 결과를 확인하고 진리표를 작성한다. 이어서 실습으로 주어진 Boolean Function (A'+B')\*C' = ((A\*B)+C)', (A'\*B')+C' = ((A+B)\*C)'역시 Verilog로 구현한 뒤 Simulation을 통해 결과를 확인하고 진리표를 작성한다. 이후 1Bit짜리 두 수의 대소를 비교하거나 일치 여부를 판별할 수 있는 1Bit비교기를 구현한 후 Simulation을 통해 제대로 구현이 되었는지를 확인하고 진리표를 작성한다. 마지막으로 최종 결과를 검토하고 관련된 추가 이론을 조사한다.

**2. De-Morgan 의 제 1,2 법칙의 simulation 결과 및 과정에 대해서 설명하시오. (NAND,NOR 과 비교 포함)**

**(1) 제 1 법칙**

- 식은 다음과 같다.

|  |
| --- |
| **De-Morgan 제1법칙** |
| = |

- Schematic은 다음과 같다.

|  |
| --- |
| **Schematic** |
|  |

- 코드는 다음과 같다.

|  |  |
| --- | --- |
| **design source** | **simulation source** |
| `timescale 1ns / 1ps  module demorganOne(ia,ib,oc,od);  input ia,ib;  output oc,od;  assign oc = ~(ia|ib);  assign od = (~ia)&(~ib);  endmodule | `timescale 1ns / 1ps  module de\_morgan\_1\_tb;  reg a,b;  wire c,d;  demorganOne u\_demorganOne(  .ia(a), .ib(b),  .oc(c), .od(d)  );  initial a = 1'b0; initial b = 1'b0;  always a = #20 ~a;  always b = #10 ~b;  initial begin  #100  $finish;  end  endmodule |

- 시뮬레이션 결과는 다음과 같다.

|  |
| --- |
| **Simulation** |
|  |

- NOR 비교

드모르간 제1법칙의 좌변과 NOR은 라는 식을 가지는 같은 게이트이다. 드 모르간(De-Morgan)의 제 1법칙은 이고, 법칙을 따르면 좌변과 우변의 Simulation결과가 같아야 한다. 실제로 같은 결과를 보이고 있으므로 법칙이 성립하는 것을 확인 할 수 있다. 따라서 드모르간 제1법칙과 NOR는 모두 같은 기능을 한다.

**(2) 제 2 법칙**

- Boolean식은 다음과 같다.

|  |
| --- |
| **De-Morgan 제2법칙** |
| = |

- Schematic은 다음과 같다.

|  |
| --- |
| **(A) Schematic** |
|  |

-코드는 다음과 같다.

|  |  |
| --- | --- |
| **design source** | **simulation source** |
| `timescale 1ns / 1ps  module demorganTwo(ia,ib,oc,od);  input ia,ib;  output oc,od;  assign oc = ~(ia&ib);  assign od = (~ia)|(~ib);  endmodule | `timescale 1ns / 1ps  module de\_morgan\_2\_tb;  reg a,b;  wire c,d;  demorganTwo u\_demorganTwo(  .ia(a), .ib(b),  .oc(c), .od(d)  );  initial a = 1'b0; initial b = 1'b0;  always a = #20 ~a;  always b = #10 ~b;  initial begin  #100  $finish;  end  endmodule |

- 시뮬레이션 결과는 다음과 같다.

|  |
| --- |
| **Simulation** |
|  |

- NAND 비교

드모르간 제2법칙의 좌변과 NAND는 라는 식을 가지는 같은 게이트이다. 드 모르간(De-Morgan)의 제 2법칙은 이고, 법칙을 따르면 좌변과 우변의 Simulation결과가 같아야 한다. 실제로 같은 결과를 보이고 있으므로 법칙이 성립하는 것을 확인 할 수 있다. 따라서 드모르간 제2법칙과 NAND는 모두 같은 기능을 한다.

-드 모르간(De-Morgan)의 정리 진리표는 다음과 같다.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |

**3. (A'+B')\*C' = ((A\*B)+C)' 의 simulation 결과 및 과정에 대해서 설명하시오. [+ 및 \* 위치 바꾼 모양도 수행]**

**(1) Boolean Function(1)**

- 식은 다음과 같다.

|  |
| --- |
| **Boolean Function(1)** |
| = |

- Schematic은 다음과 같다.

|  |
| --- |
| **Schematic** |
|  |

-코드는 다음과 같다.

|  |  |
| --- | --- |
| **design source** | **simulation source** |
| `timescale 1ns / 1ps  module booleanFuncOne(ia,ib,ic,od,oe);  input ia,ib,ic;  output od,oe;  assign od = ((~ia)|(~ib))&(~ic);  assign oe = ~((ia&ib)|ic);  endmodule | `timescale 1ns / 1ps  module boolean\_function\_1\_tb;  reg a,b,c;  wire d,e;  booleanFuncOne u\_booleanFuncOne(  .ia(a), .ib(b), .ic(c),  .od(d), .oe(e)  );  initial a = 1'b0; initial b = 1'b0; initial c = 1'b0;  always a = #20 ~a;  always b = #10 ~b;  always c = #5 ~c;  initial begin  #100  $finish;  end  endmodule |

- 시뮬레이션 결과는 다음과 같다.

|  |
| --- |
| **Simulation** |
|  |

Simulation을 통해 좌변과 우변의 동작이 같은 것을 확인할 수 있다. 이는 드모르간 법칙으로도 확인할 수 있다.

**(2) Boolean Function(2)**

- Boolean식은 다음과 같다.

|  |
| --- |
| **Boolean Function(2)** |
| = |

- Schematic은 다음과 같다.

|  |
| --- |
| **Schematic** |
|  |

-코드는 다음과 같다.

|  |  |
| --- | --- |
| **design source** | **simulation source** |
| `timescale 1ns / 1ps  module booleanFuncTwo(ia,ib,ic,od,oe);  input ia,ib,ic;  output od,oe;  assign od = ((~ia)&(~ib))|(~ic);  assign oe = ~((ia|ib)&ic);  endmodule | `timescale 1ns / 1ps  module boolean\_function\_2\_tb;  reg a,b,c;  wire d,e;  booleanFuncTwo u\_booleanFuncTwo(  .ia(a), .ib(b), .ic(c),  .od(d), .oe(e)  );  initial a = 1'b0; initial b = 1'b0; initial c = 1'b0;  always a = #20 ~a;  always b = #10 ~b;  always c = #5 ~c;  initial begin  #100  $finish;  end  endmodule |

- 시뮬레이션 결과는 다음과 같다.

|  |
| --- |
| **Simulation** |
|  |

Simulation을 통해 좌변과 우변의 동작이 같은 것을 확인할 수 있다. 이는 드모르간 법칙으로도 확인할 수 있다.

- Boolean Function의 진리표는 다음과 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **(A'+B')\*C'** | **((A\*B)+C)'** | **(A'\*B')+C'** | **((A+B)\*C)'** |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 |

**4. 1Bit 비교기의 simulation 결과 및 과정에 대해서 설명하시오.(2 input, 4 output)[진리표 작성]**

- 다음과 같은 식 4개를 구현한다.

|  |  |
| --- | --- |
| **a == b** | **a != b** |
| **a > b** | **a < b** |

- Schematic은 다음과 같다.

|  |
| --- |
| **Schematic** |
|  |

- 코드는 다음과 같다.

|  |  |
| --- | --- |
| **design source** | **simulation source** |
| `timescale 1ns / 1ps  module oneBitCompare(ia,ib,oc,od,oe,of);  input ia,ib;  output oc,od,oe,of;  assign oc = ia==ib;  assign od = ia!=ib;  assign oe = ia>ib;  assign of = ia<ib;  endmodule | `timescale 1ns / 1ps  module one\_bit\_tb;  reg a,b;  wire c,d,e,f;  oneBitCompare u\_oneBitCompare(  .ia(a), .ib(b),  .oc(c), .od(d), .oe(e), .of(f)  );  initial a = 1'b0; initial b = 1'b0;  always a = #20 ~a;  always b = #10 ~b;  initial begin  #100  $finish;  end  endmodule |

- 시뮬레이션 결과는 다음과 같다.

|  |
| --- |
| **(A) Simulation** |
|  |

c는 입력 a와b가 같은 값일 때 1을 나타내고 그렇지 않을 때 0을 출력한다. d는 c와 반대로 두 입력이 같은 값일 때 0, 다른 값일 때 1을 출력한다. e는 a가 b보다 클 때, f는 b가 a보다 클 때 1을 출력한다. Simulation결과를 통해 제대로 작동함을 확인할 수 있다.

- 진리표는 다음과 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c(a==b)** | **d(a!=b)** | **e(a>b)** | **f(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 |

**5. 결과 검토 및 논의사항.**

논리합의 전체 부정을 각 변수에 부정을 취한 논리곱으로 바꿀 수 있는 De-Morgan 제 1법칙과 논리곱의 전체 부정을 각 변수에 부정을 취한 논리합으로 바꿀 수 있는 De-Morgan 제 2법칙을 살펴보았다. 와 를 Verilog로 구현하여 Simulation한 결과 제 1법칙이 성립하는 것을 확인 할 수 있었고 마찬가지로 와 또한 Simulation을 통해 결과를 확인하여 제 2법칙이 성립하는 것을 증명하였다. 이후 임의의 Boolean function인 와 , 와 를 코드로 구현하여 Simulation을 하였고, 결과적으로 각각 같은 기능을 하는 function임을 증명하였다. 이는 앞에서 살펴본 De-Morgan법칙을 통해서도 증명이 가능한 것을 확인할 수 있었다. 끝으로 1Bit 비교기 또한 구현함으로써 관계연산자를 사용하였고, Simulation을 통해 (a==b, a!=b, a>b, a<b)총 4개의 동작을 제대로 구현하는 것을 확인할 수 있었다.

Simulation을 통해 단순히 한 function이 어떤 결과를 보이는지를 확인할 수도 있지만 어떠한 식이 성립하는 식인지, 어떠한 공식이 정말 맞는지 아닌지 Simulation 결과를 통해 증명할 수 있음을 알게 되었다. 만일 임의의 boolean식을 다양한 법칙을 사용하여 간소화를 했다면 Simulation을 통해 제대로 간소화를 한 건지 확인해 볼 수 있을 것이다.

**6. 추가 이론 조사 및 작성**

- SOP와 POS

SOP란 Sum Of Product의 약자로 boolean식의 변수들을 곱하고 이 항들을 모두 더한 형태를 의미한다. POS는 Product Of Sum의 약자로 boolean식의 변수들을 합하고 항들을 모두 곱한 형태를 의미한다. 그 중 f(a,b,c) = abc + a’bc’ + a’b’c 또는 f(a,b,c) = (a+b+c)(a’+b+c)(a’+b’+c’)와 같이 각 항에 빠짐없이 모든 종류의 변수가 들어가 있는 형태를 Canonical form이라고 한다. 진리표를 보고 Canonical form인 SOP 또는 POS로 boolean식을 구할 수 있다. 예를 들면 다음과 같은 진리표를 보이는 식이 있다.

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **F** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

F의 SOP를 구하기 위해서는 F의 출력값이 1인 부분을 골라 결과에 대응하는 입력 (A,B,C)를 곱한 이후 만든 항들을 모두 더해주면 된다.

F = A’BC + AB’C + ABC’ + ABC

F의 POS를 구하기 위해서는 F의 출력값이 0인 부분을 골라 결과에 대응하는 입력 (A,B,C)를 더한 이후 만들어진 항을 모두 곱해주거나(이때 SOP와 달리 입력값이 1일 때 부정인 형태로 변수를 작성해야 한다.)F’의 SOP형태를 구해주고 그 이후 드모르간 법칙을 사용하면 된다. 다음은 F’의 SOP형태를 구한 뒤 드모르간 법칙을 사용한 결과이다.

F’ = A’B’C’ + A’B’C + A’BC’ + AB’C’

(F’)’ = (A’B’C’ + A’B’C + A’BC’ + AB’C’)’ = (A+B+C)(A+B+C’)(A+B’+C)(A’+B+C)