**9주차 결과 보고서**

남민혁

**1. 2 to 4 Decoder 의 결과 및 Simulation 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함,[AND GATE])**

**-Truth Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | | **Output** | | | |
| **A** | **B** | **D0** | **D1** | **D2** | **D3** |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

**- Karnaugh Map (D0)**

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

**- Karnaugh Map (D1)**

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

**- Karnaugh Map (D2)**

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

**- Karnaugh Map (D3)**

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

**-최소화된 Boolean 함수**

위의 Karnaugh map을 활용하여 각 output에 대하여 최소화한 식은 다음과 같다.

**-Verilog 코딩**

다음은 최소화한 식을 바탕으로 Verilog 코드를 작성한 결과이다. 각각은 논리회로 구성을 위한 Verilog 코드, 이를 바탕으로 모든 가능한 입력으로 부터 기대되는 출력이 나타나는지를 확인하기 위한 Test bench 코드이다.

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module two\_to\_four\_decoder(  input a,  input b,  output d0,  output d1,  output d2,  output d3  );  assign d0 = ~a & ~b;  assign d1 = ~a & b;  assign d2 = a & ~b;  assign d3 = a & b;  endmodule |
| Test Bench | `timescale 1ns / 1ps  module two\_to\_four\_decoder\_tb;  reg aa;  reg bb;  wire dd0;  wire dd1;  wire dd2;  wire dd3;  two\_to\_four\_decoder u\_two\_to\_four\_decoder(  .a (aa),  .b (bb),  .d0 (dd0),  .d1 (dd1),  .d2 (dd2),  .d3 (dd3)  );  initial aa = 1'b0;  initial bb = 1'b0;  always aa = #200 ~ aa;  always bb = #100 ~bb;  initial begin  #400  $finish;  end  endmodule |

**-Simulation 출력 결과 비교**

**A picture containing green, sitting, screenshot, computer

Description automatically generated**

모든 가능한 입력 값에 대한 출력값을 확인하기 위하여 Simulation을 시행한 결과 의도한 진리표와 모든 결과가 동일함을 확인하였다.

**-구현된 Schematic**

Diagram, schematic

Description automatically generated

**2. 4 to 2 Encoder 의 결과 및 Simulation 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)**

**-Truth Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | | | | **Output** | |
| **A** | **B** | **C** | **D** | **E0** | **E1** |
| 0 | 0 | 0 | 0 | X | X |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | X | X |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | X | X |
| 0 | 1 | 1 | 0 | X | X |
| 0 | 1 | 1 | 1 | X | X |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | X | X |
| 1 | 0 | 1 | 0 | X | X |
| 1 | 0 | 1 | 1 | X | X |
| 1 | 1 | 0 | 0 | X | X |
| 1 | 1 | 0 | 1 | X | X |
| 1 | 1 | 1 | 0 | X | X |
| 1 | 1 | 1 | 1 | X | X |

**- Karnaugh Map (E0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CD  AB | 00 | 01 | 11 | 10 |
| 00 | X | 1 | X | 1 |
| 01 | 0 | X | X | X |
| 11 | X | X | X | X |
| 10 | 0 | X | X | X |

**- Karnaugh Map (E0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CD  AB | 00 | 01 | 11 | 10 |
| 00 | X | 1 | X | 0 |
| 01 | 1 | X | X | X |
| 11 | X | X | X | X |
| 10 | 0 | X | X | X |

**-최소화된 Boolean 함수**

위의 Karnaugh map을 활용하여 각 output에 대하여 최소화한 식은 다음과 같다.

**-Verilog 코딩**

다음은 최소화한 식을 바탕으로 Verilog 코드를 작성한 결과이다. 각각은 논리회로 구성을 위한 Verilog 코드, 이를 바탕으로 모든 가능한 입력으로 부터 기대되는 출력이 나타나는지를 확인하기 위한 Test bench 코드이다.

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module four\_to\_two\_encoder(  input a,  input b,  input c,  input d,  output e0,  output e1  );  assign e0 = c | d;  assign e1 = b | d;  endmodule |
| Test Bench | `timescale 1ns / 1ps  module four\_to\_two\_encoder\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  wire ee0;  wire ee1;  four\_to\_two\_encoder u\_four\_to\_two\_encoder(  .a (aa),  .b (bb),  .c (cc),  .d (dd),  .e0 (ee0),  .e1 (ee1)  );  initial aa = 1'b0;  initial bb = 1'b0;  initial cc = 1'b0;  initial dd = 1'b0;  always aa = #400 ~ aa;  always bb = #200 ~bb;  always cc = #100 ~cc;  always dd = #50 ~dd;  initial begin  #800  $finish;  end  endmodule |

**-Simulation 출력 결과 비교**

**Graphical user interface

Description automatically generated**

모든 가능한 입력 값에 대한 출력값을 확인하기 위하여 Simulation을 시행한 결과 의도한 진리표와 모든 결과가 동일함을 확인하였다.

**-구현된 Schematic**

Diagram

Description automatically generated

**3. BCD to Decimal decoder 의 결과 및 Simulation 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)**

**-Truth Table**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | | | | **Output** | | | | | | | | | |
| **A3** | **A2** | **A1** | **A0** | **O0** | **O1** | **O2** | **O3** | **O4** | **O5** | **O6** | **O7** | **O8** | **O9** |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**- Karnaugh Map (O0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A1A0  A3A2 | 00 | 01 | 11 | 10 |
| 00 | 1 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

**- Karnaugh Map (O1)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A1A0  A3A2 | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

**- Karnaugh Map (O2)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A1A0  A3A2 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 1 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

**- Karnaugh Map (O3)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A1A0  A3A2 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

**- Karnaugh Map (O4)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A1A0  A3A2 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

**- Karnaugh Map (O5)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A1A0  A3A2 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

**- Karnaugh Map (O6)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A1A0  A3A2 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 1 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

**- Karnaugh Map (O7)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A1A0  A3A2 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

**- Karnaugh Map (O8)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A1A0  A3A2 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 0 |

**- Karnaugh Map (O9)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A1A0  A3A2 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 |

**-최소화된 Boolean 함수**

위의 Karnaugh map을 활용하여 각 output에 대하여 최소화한 식은 다음과 같다.

**-Verilog 코딩**

다음은 최소화한 식을 바탕으로 Verilog 코드를 작성한 결과이다. 각각은 논리회로 구성을 위한 Verilog 코드, 이를 바탕으로 모든 가능한 입력으로 부터 기대되는 출력이 나타나는지를 확인하기 위한 Test bench 코드이다.

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module bcd\_to\_dec\_decoder(  input a0,  input a1,  input a2,  input a3,  output o0,  output o1,  output o2,  output o3,  output o4,  output o5,  output o6,  output o7,  output o8,  output o9  );  assign o0= ~a0 & ~a1 & ~a2 & ~a3;  assign o1= a0 & ~a1 & ~a2 & ~a3;  assign o2 = ~a0 & a1 & ~a2 & ~a3;  assign o3 = a0 & a1 & ~a2 & ~a3;  assign o4 = ~a0 & ~a1 & a2 & ~a3;  assign o5 = a0 & ~a1 & a2 & ~a3;  assign o6 = ~a0 & a1 & a2 & ~a3;  assign o7 = a0 & a1 & a2 & ~a3;  assign o8 = ~a0 & ~a1 & ~a2 & a3;  assign o9 = a0 & ~a1 & ~a2 & a3;  endmodule |
| Test Bench | `timescale 1ns / 1ps  module bcd\_to\_dec\_decoder\_tb;  reg aa0;  reg aa1;  reg aa2;  reg aa3;  wire oo0;  wire oo1;  wire oo2;  wire oo3;  wire oo4;  wire oo5;  wire oo6;  wire oo7;  wire oo8;  wire oo9;  bcd\_to\_dec\_decoder u\_bcd\_to\_dec\_decoder(  .a0 (aa0),  .a1 (aa1),  .a2 (aa2),  .a3 (aa3),  .o0 (oo0),  .o1 (oo1),  .o2 (oo2),  .o3 (oo3),  .o4 (oo4),  .o5 (oo5),  .o6 (oo6),  .o7 (oo7),  .o8 (oo8),  .o9 (oo9)  );  initial aa0 = 1'b0;  initial aa1 = 1'b0;  initial aa2 = 1'b0;  initial aa3 = 1'b0;  always aa0 = #50 ~aa0;  always aa1 = #100 ~aa1;  always aa2 = #200 ~aa2;  always aa3 = #400 ~aa3;  initial begin  #500  $finish;  end  endmodule |

**-Simulation 출력 결과 비교**

**Chart

Description automatically generated**

모든 가능한 입력 값에 대한 출력값을 확인하기 위하여 Simulation을 시행한 결과 의도한 진리표와 모든 결과가 동일함을 확인하였다.

**-구현된 Schematic**

Diagram, schematic

Description automatically generated

**4. 4 to 1 line MUX의 결과 및 Simulation 과정에 대해서 설명하시오. (code, Truth table 작성)**

**-Truth Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Select** | | **Input** | | | | **Output** |
| **S1** | **S0** | **A** | **B** | **C** | **D** | **Out** |
| 0 | 0 | 0 | X | X | X | 0 |
| 0 | 0 | 1 | X | X | X | 1 |
| 0 | 1 | X | 0 | X | X | 0 |
| 0 | 1 | X | 1 | X | X | 1 |
| 1 | 0 | X | X | 0 | X | 0 |
| 1 | 0 | X | X | 1 | X | 1 |
| 1 | 1 | X | X | X | 0 | 0 |
| 1 | 1 | X | X | X | 1 | 1 |

**- Karnaugh Map (S0=0, S1=0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CD  AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

**- Karnaugh Map (S0=1, S1=0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CD  AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

**- Karnaugh Map (S0=0, S1=1)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CD  AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 1 | 1 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | 0 | 0 | 1 | 1 |
| 10 | 0 | 0 | 1 | 1 |

**- Karnaugh Map (S0=1, S1=1)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CD  AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 1 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 0 | 1 | 1 | 0 |

**-최소화된 Boolean 함수**

위의 Karnaugh map을 활용하여 각 output에 대하여 최소화한 식은 다음과 같다.

**-Verilog 코딩**

다음은 최소화한 식을 바탕으로 Verilog 코드를 작성한 결과이다. 각각은 논리회로 구성을 위한 Verilog 코드, 이를 바탕으로 모든 가능한 입력으로 부터 기대되는 출력이 나타나는지를 확인하기 위한 Test bench 코드이다.

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module four\_to\_one\_mux(  input a,  input b,  input c,  input d,  input s0,  input s1,  output out  );  assign out = (a & ~s1 & ~s0) | (b & ~s1 & s0) | (c & s1 & ~s0) | (d & s1 & s0);  endmodule |
| Test Bench | `timescale 1ns / 1ps  module four\_to\_one\_mux\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  reg ss0;  reg ss1;  wire out;  four\_to\_one\_mux u\_four\_to\_one\_mux(  .a (aa),  .b (bb),  .c (cc),  .d (dd),  .s0 (ss0),  .s1 (ss1),  .out (out)  );  initial ss0 = 1'b0;  initial ss1 = 1'b0;  initial aa = 1'b0;  initial bb = 1'b0;  initial cc = 1'b0;  initial dd = 1'b0;  always ss1 = #320 ~ss1;  always ss0 = #160 ~ss0;  always aa = #80 ~aa;  always bb = #40 ~bb;  always cc = #20 ~cc;  always dd = #10 ~dd;  initial begin  #640  $finish;  end  endmodule |

**-Simulation 출력 결과 비교**

**Graphical user interface

Description automatically generated**

모든 가능한 입력 값에 대한 출력값을 확인하기 위하여 Simulation을 시행한 결과 의도한 진리표와 모든 결과가 동일함을 확인하였다.

**-구현된 Schematic**

Diagram, schematic

Description automatically generated

**5. 1 to 4 line deMUX를 이용하여 2 to 4 decoder와 어떤 관계인지 간략하게 설명하시오.**

1 to 4 line deMUX는 2개의 select 비트에 따라서 4개 중 한 군데의 출력 단에 데이터 입력을 보내는 논리 회로이다. 반면, 2 to 4 decoder는 2개의 입력을 바탕으로 4개의 출력 중 하나의 출력에만 1을 가지는 코드로 해독해주는 논리회로이다.

두 논리 회로의 사용 방식은 다르나, 다음 진리표에서 볼 수 있듯이, 1 to 4 line deMUX의 Data가 1일 때, select비트가 곧 2 to 4 decoder에서 두개의 입력과 같은 역할을 한다고 볼 수 있다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Select** | | **Data** | **Input** | | | |
| **S1** | **S0** | **A** | **D0** | **D1** | **D2** | **D3** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | | **Output** | | | |
| **A** | **B** | **D0** | **D1** | **D2** | **D3** |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

**6. 추가 이론 조사 및 작성.**

Decoder는 N개의 입력에 대해 가능한 모든 논리조합 2의 N승 개의 출력 종류에 대해 출력을 하는 Minterm (AND 게이트에 대해서), 또는 Maxterm (NAND 게이트에 대해) 로서, 각 출력에 대해 다음과 같이 고려할 수 있다.

A picture containing diagram

Description automatically generated

따라서 특정 출력이 mi, mj, mk의 논리곱인 경우 해당 출력을 AND게이트로, 반대로 논리합인 경우에는 OR 게이트로 decoder의 출력 결과를 활용함으로써, 동일 출력을 구현할 수 있다. 이 같은 방식은 decoder가 필요없는 min/max-term까지 만듦으로써 필요없는 연산을 진행하는 등 논리 회로를 최소화하여 최적화하지는 못하지만, 원하는 출력의 논리 식을 가지고 있는 경우 빠르게 만들 수 있다는 장점이 있다.

예를 들어 두개의 입력 (x, y)와 하나의 carry bit (z)를 활용한 Full adder에서 Sum은 1, 2, 4, 7 miniterm의 논리합, Carry는 3, 5, 6, 7 miniterm의 논리합이므로, 다음과 같이 구현할 수 있다.

Diagram

Description automatically generated

**7. 참고 문헌**

강석태, “Verilog HDL Summary”, <http://vlsi.hongik.ac.kr/lecture/%EC%8B%A4%ED%97%98/Verilog_Summary.pdf>.

엠에스리, “엔코더 (Encoder), 디코더 (Decoder)”, <https://m.blog.naver.com/lagrange0115/220717414549>.

엠에스리, “멀티플렉서 (Multiplexer), 디멀티플렉서 (Demultiplexer)”, <https://m.blog.naver.com/lagrange0115/220716574603>

위키피디아, “Priority Encoder”, <https://en.wikipedia.org/wiki/Priority_encoder>.

위키피디아, “Encoder (digital)”, <https://en.wikipedia.org/wiki/Simple_encoder>.

위키피디아, “Multiplexer”, <https://en.wikipedia.org/wiki/Multiplexer>.

한국기술교육대학교 전기전자통신공학부, “디지털시스템 설계 및 실습” Verilog HDL 문법 강의자료, <https://cms3.koreatech.ac.kr/sites/yjjang/down/dsys11/M01_VerilogHDL01.pdf>.