**7주차 결과 보고서**

남민혁

**1. 실험 목적**

7주차 실험의 목적은 아래와 같다.

* Parity bit Generator/Checker 및 2bit binary comparator 의 동작의 이해 및 확인
* Verilog를 사용하여 Parity bit Generator/Checker , 2bit binary comparator 구현
* 입력 신호 생성 후 Simulation을 통하여 구현된 각 Gate 동작 확인

**2. Even Parity bit generator 및 checker 의 simulation 결과 및 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)**

**1) Even Parity Bit Generator**

**-Truth Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | | Output |
| A | B | C | D | E |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

**- Karnaugh Map**

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

**-최소화된 Boolean 함수**

**-Verilog 코딩**

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module even\_parity\_bit\_generator(  input a,  input b,  input c,  input d,  output e  );  assign e = a ^ b ^ c ^ d;  endmodule |
| Test Bench | `timescale 1ns / 1ps  module even\_parity\_bit\_generator\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  wire ee;  even\_parity\_bit\_generator u\_even\_parity\_bit\_generator(  .a (aa),  .b (bb),  .c (cc),  .d (dd),  .e (ee)  );  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**

**-구현된 Schematic**

Diagram

Description automatically generated

**2) Even Parity Bit Checker**

**-Truth Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | | | | Output |
| A | B | C | D | P | PEC |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

**- Karnaugh Map (A=0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DP  BC | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 0 | 1 |
| 01 | 1 | 0 | 1 | 0 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |

**- Karnaugh Map (A=1)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DP  BC | 00 | 01 | 11 | 10 |
| 00 | 1 | 0 | 1 | 0 |
| 01 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 |

**-최소화된 Boolean 함수**

**-Verilog 코딩**

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module even\_parity\_bit\_checker(  input a,  input b,  input c,  input d,  input p,  output pec  );  assign pec = a ^ b ^ c ^ d ^ p;  endmodule |
| Test Bench | `timescale 1ns / 1ps  module even\_parity\_bit\_checker\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  reg pp;  wire pecc;  even\_parity\_bit\_checker u\_even\_parity\_bit\_checker(  .a (aa),  .b (bb),  .c (cc),  .d (dd),  .p (pp),  .pec (pecc)  );  initial aa = 1'b0;  initial bb = 1'b0;  initial cc = 1'b0;  initial dd = 1'b0;  initial pp = 1'b0;  always aa = #400 ~ aa;  always bb = #200 ~bb;  always cc = #100 ~cc;  always dd = #50 ~dd;  always pp = #25 ~pp;  initial begin  #800  $finish;  end  endmodule |

**-Simulation 출력 결과 비교**

**Graphical user interface

Description automatically generated**

**-구현된 Schematic**

Diagram

Description automatically generated

**3. Odd Parity bit generator 및 checker 의 simulation 결과 및 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)**

**1) Odd Parity Bit Generator**

**-Truth Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | | Output |
| A | B | C | D | E |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

**- Karnaugh Map**

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

**-최소화된 Boolean 함수**

**-Verilog 코딩**

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module odd\_parity\_bit\_generator(  input a,  input b,  input c,  input d,  output e  );  assign e = ~(a ^ b ^ c ^ d);  endmodule |
| Test Bench | `timescale 1ns / 1ps  module odd\_parity\_bit\_generator\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  wire ee;  odd\_parity\_bit\_generator u\_odd\_parity\_bit\_generator(  .a (aa),  .b (bb),  .c (cc),  .d (dd),  .e (ee)  );  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**

**-구현된 Schematic**

Diagram

Description automatically generated

**2) Odd Parity Bit Checker**

**-Truth Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | | | | Output |
| A | B | C | D | P | PEC |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

**- Karnaugh Map (A=0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DP  BC | 00 | 01 | 11 | 10 |
| 00 | 1 | 0 | 1 | 0 |
| 01 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 |

**- Karnaugh Map (A=1)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DP  BC | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 0 | 1 |
| 01 | 1 | 0 | 1 | 0 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |

**-최소화된 Boolean 함수**

**-Verilog 코딩**

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module odd\_parity\_bit\_checker(  input a,  input b,  input c,  input d,  input p,  output pec  );  assign pec = ~(a ^ b ^ c ^ d ^ p);  endmodule |
| Test Bench | `timescale 1ns / 1ps  module odd\_parity\_bit\_checker\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  reg pp;  wire pecc;  odd\_parity\_bit\_checker u\_odd\_parity\_bit\_checker(  .a (aa),  .b (bb),  .c (cc),  .d (dd),  .p (pp),  .pec (pecc)  );  initial aa = 1'b0;  initial bb = 1'b0;  initial cc = 1'b0;  initial dd = 1'b0;  initial pp = 1'b0;  always aa = #400 ~ aa;  always bb = #200 ~bb;  always cc = #100 ~cc;  always dd = #50 ~dd;  always pp = #25 ~pp;  initial begin  #800  $finish;  end  endmodule |

**-Simulation 출력 결과 비교**

**Graphical user interface

Description automatically generated**

**-구현된 Schematic**

Diagram

Description automatically generated

**4. 2-bit binary comparator simulation 결과 및 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)**

**-Truth Table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | | | | | **Output** | | | |
| A1 | A2 | B1 | B2 | F1(>) | | F2(=) | F3(<) |
| 0 | 0 | 0 | 0 | 0 | | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | | 0 | 1 |
| 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 | 0 | | 1 | 0 |

**- Karnaugh Map (F1)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B1B2  A1A2 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 11 | 1 | 1 | 0 | 1 |
| 10 | 1 | 1 | 0 | 0 |

**- Karnaugh Map (F2)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B1B2  A1A2 | 00 | 01 | 11 | 10 |
| 00 | 1 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 |
| 10 | 0 | 0 | 0 | 1 |

**- Karnaugh Map (F3)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B1B2  A1A2 | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 1 | 1 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 |

**-최소화된 Boolean 함수**

**-Verilog 코딩**

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module two\_bits\_binary\_comparator(  input a1,  input a2,  input b1,  input b2,  output f1,  output f2,  output f3  );  assign f1 = (a1 & ~b1) | (a2 & ~b1 & ~b2) | (a1 & a2 & ~b2); /\*greater than\*/  assign f2 = (a1 ^~ b1) & (a2 ^~ b2); /\*equal\*/  assign f3 = (~a1 & b1) | (~a1 & ~a2 & b2) | (~a2 & b1 & b2); /\*less than\*/  endmodule |
| Test Bench | `timescale 1ns / 1ps  module two\_bits\_binary\_comparator\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  wire f11;  wire f22;  wire f33;  two\_bits\_binary\_comparator u\_two\_bits\_binary\_comparator(  .a1 (aa),  .a2 (bb),  .b1 (cc),  .b2 (dd),  .f1 (f11),  .f2 (f22),  .f3 (f33)  );  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**

**A picture containing green, sitting, monitor, screen

Description automatically generated**

**-구현된 Schematic**

Diagram, schematic

Description automatically generated

**5. 결과 검토 및 논의사항**

이번주 실험에서 짝수, 홀수Parity Bit 생성기와 검사기, 그리고 2비트 비교기를 구현하였다. 구현을 위하여 주어진 논리에 대하여 진리표를 구성한 후, 이를 활용하여 Karnaugh Map를 이용하여 SOP와 POS 형태로 최소화 하였다. 그러나 Parity Bit 생성기와 검사기의 경우에는 Karnaugh Map으로는 최소화가 어려워, Parity Bit의 생성 원리(총 input에서의 1의 개수가 홀수나 짝수개여야 한다는 사실)를 활용하여 최소화하였다. 끝으로 최소화한 식을 Verilog 코드로 작성하고 모든 가능한 입력에 대해 simulate 해보았다. 그 결과 모든 디자인이 의도한 진리표와 다르지 않고 동일하게 결과가 출력됨을 확인하였다.

**6. 추가 이론 조사 및 작성**

구현한 2비트 비교기는 이론적으로 N 비트 비교기로 확장할 수 있으며, 현실에서는 4비트 비교기를 여러 개 연결하여 구현한다. 4비트 비교기는 4비트로 표현된 2개의 수 즉 A3-A0와 B3-B0의 크기를 비교하는 회로이다. 따라서 8개의 입력을 받고, 대소관계와 동일한지 여부를 판단하기 위한 3개의 출력으로 구성된 논리회로이다. 구체적으로 A>B이면 가 1, A<B이면 가 1, A=B이면 가 1을 출력한다.

4비트 비교기는 MSB(Most Significant Bit), 즉 자릿수가 가장 큰 비트 (현재 예시에서는 하첨자 3을 가진 비트) 부터 비교하여 동일하지 않으면, 해당 대소관계가 곧 받은 입력 전체의 대소관계로 확정지어진다. 예를 들어 1000과 0100을 비교한다고 하면, MSB가 1과 0이므로, 1을 가진 1000가 0100보다 크다고 결정짓는 방식으로 구현된다. 구체적으로 진리표와 이를 위한 논리회로는 다음과 같다.

![A close up of a keyboard

Description automatically generated]() ![Diagram

Description automatically generated]()

4개의 입력값 외에 진리표에 표시된 입력은 4비트 비교기를 활용하여, 4비트 이상의 N비트 비교기로 확장하기 위하여 앞선 상위 비트의 비교 결과를 다음 7485비교기에 알려주기 위함이다.

이러한 비교기는 가산기와 더불어 정보처리의 기본으로서 CPU 와 MCU에 구현되어 있다. 그 밖에도 다음과 같은 다양한 분야에서 활용되고 있다. 비밀번호 검증이나, 물리적인 변수들을 기준 값과 비교해야하는 분야에서 많이 사용된다.

**7. 참고 문헌**

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

위키피디아, “Digital comparator”, <https://en.wikipedia.org/wiki/Digital_comparator>.

장영조, “디지털공학 및 실습 7장 강의자료: 조합 논리 회로”, <https://cms3.koreatech.ac.kr/sites/yjjang/down/dig_new/chap7.pdf>.

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