컴퓨터공학실험 2 7 주차 결과보고서

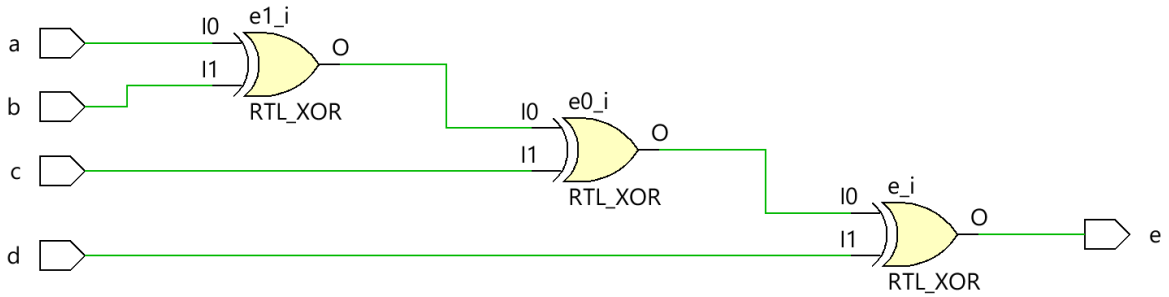
전공: 컴퓨터공학과 학년: 2 학번: 20191559 이름: 강상원

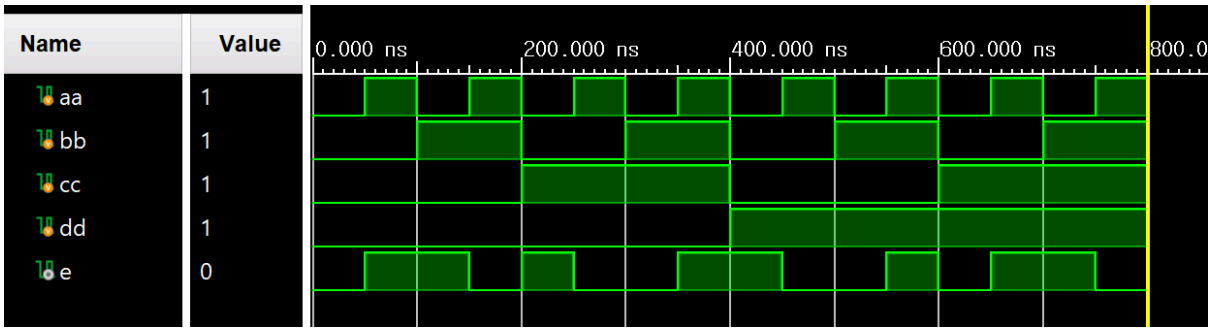
1. Even Parity bit generator 및 checker의 simulation 결과 및 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)

- Even Parity bit generator

  assign e = a ^ b ^ c ^ d와 같이 나타낼 수 있다. (간소화 방법은 후술) Schematic과 Simulation 결과는 아래와 같다.

```
1   `timescale 1ns / 1ps
2   module parity_bit_gen(
3       input a, b, c, d,
4       output e
5       );
6       assign e = a ^ b ^ c ^ d;
7
8   endmodule
9
```





e는 a, b, c, d를 모두 XOR 한 값이기 때문에 a~d 중 1의 개수가 홀수여야 1이 반환된다.

➔ 1의 개수가 홀수이면, parity bit이 1이 되어 even parity를 맞춘다.

↓ 진리표

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

↓ 카르노 맵

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

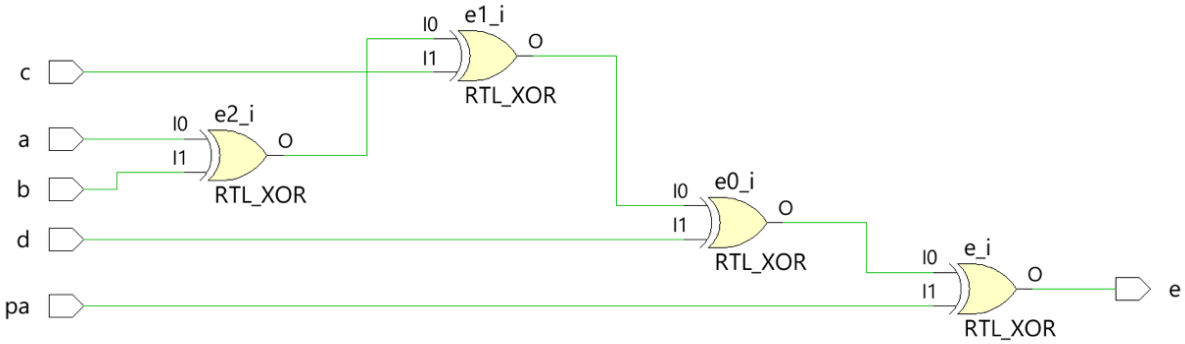
위 카르노 맵을 정리하면, $e = a \oplus b \oplus c \oplus d$로 나타낼 수 있다.

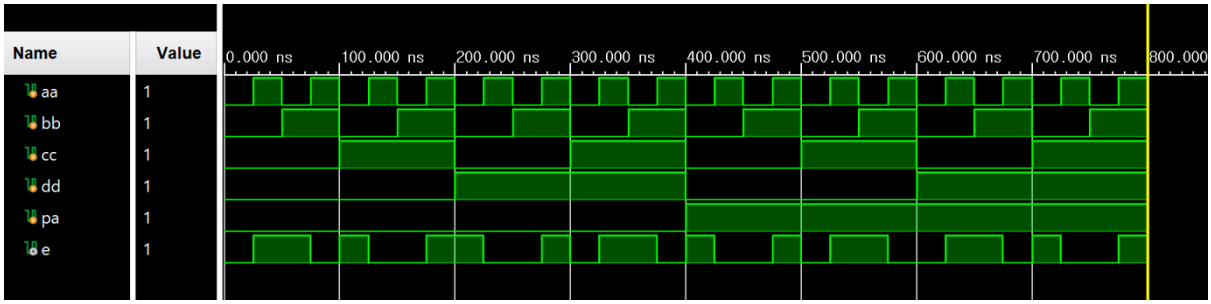
∴ assign e = a ^ b ^ c ^ d와 같이 나타낼 수 있다.

- Even Parity bit checker

  assign e = a ^ b ^ c ^ d ^ pa와 같이 나타낼 수 있다. (간소화 방법은 후술) Schematic과 Simulation 결과는 아래와 같다.

```
1       `timescale 1ns / 1ps
2       module parity_bit_check(
3           input a, b, c, d, pa,
4           output e
5           );
6           assign e = a ^ b ^ c ^ d ^ pa;
7
8       endmodule
9
```





Even parity여야 하므로, a~d + parity bit 중 1의 개수가 홀수이면 error 값이 1이 된다. (오류 검출)

(진리표 다음쪽에)

↓ 진리표

a	b	c	d	parity	e
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	1
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	1	1
1	0	1	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	0
1	1	1	0	0	1
1	1	1	0	1	0
1	1	1	1	0	0
1	1	1	1	1	1

parity=0

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

parity=1

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

위 카르노 맵을 정리하면, $e = a \oplus b \oplus c \oplus d \oplus parity$로 나타낼 수 있다.

∴ assign e = a ^ b ^ c ^ d ^ pa와 같이 나타낼 수 있다.

2. Odd Parity bit generator 및 checker의 simulation 결과 및 과정에 대해서 설명하시오. (Truth table 작성 및 k-map 포함)

- **Odd Parity bit generator**

Odd Parity bit generator 는 a, b, c, d 중 1 의 개수가 짝수이면, parity bit 이 1 이 되어 Odd parity 를 맞춘다.

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

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

따라서 논리식은 $e = \overline{a \oplus b \oplus c \oplus d}$와 같다.

∴ assign e = ~ (a ^ b ^ c ^ d)와 같이 나타낼 수 있다.

Verilog 코딩과 시뮬레이션 결과는 Even Parity bit generator 의 정반대라 볼 수 있다.

- **Odd Parity bit checkor**

Odd parity여야 하므로, a~d + parity bit 중 1의 개수가 짝수이면 error 값이 1이 된다. (오류 검출)

a	b	c	d	parity	e
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	1	1
0	0	1	1	0	1
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	0	1	1
0	1	0	1	0	1
0	1	0	1	1	0
0	1	1	0	0	1
0	1	1	0	1	0
0	1	1	1	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	1
1	0	0	1	0	1
1	0	0	1	1	0
1	0	1	0	0	1

1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	0	1	1
1	1	1	1	0	1
1	1	1	1	1	0

↓ 카르노 맵

parity=0

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

parity=1

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

위 카르노 맵을 정리하면, $e = \overline{\text{a}\oplus\text{b}\oplus\text{c}\oplus\text{d}\oplus parity}$로 나타낼 수 있다.

∴ assign e = ~ (a ^ b ^ c ^ d ^ pa)와 같이 나타낼 수 있다.

Verilog 코딩과 시뮬레이션 결과는 Even Parity bit checker 의 정반대라 볼 수 있다.

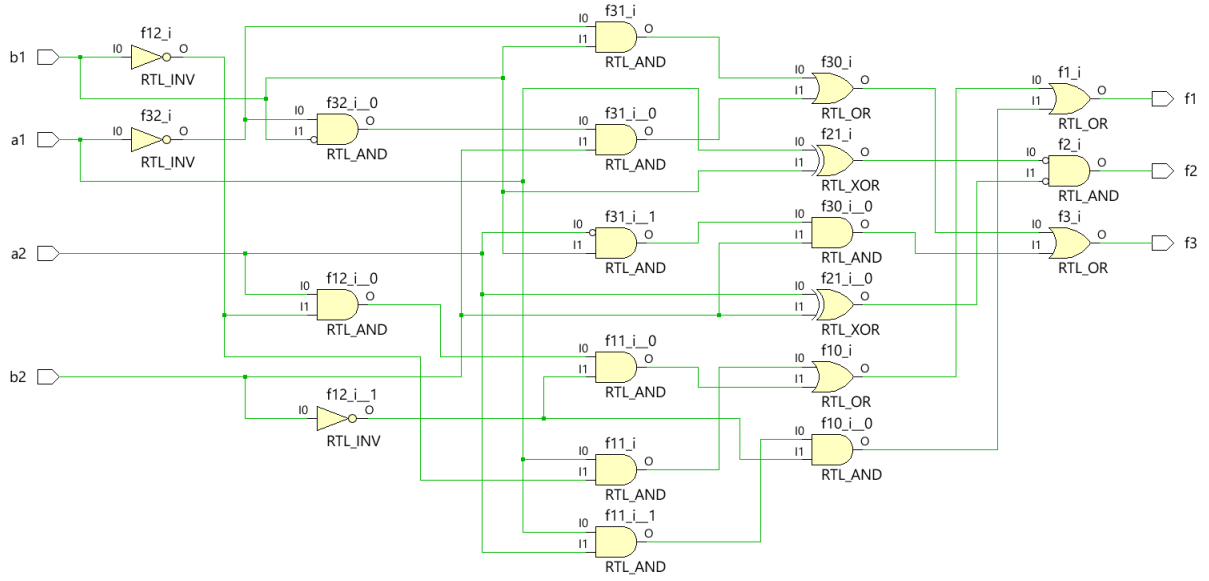
3. 2-bit binary comparator simulation 결과 및 과정에 대해서 설명하시오.
(Truth table 작성 및 k-map 포함)

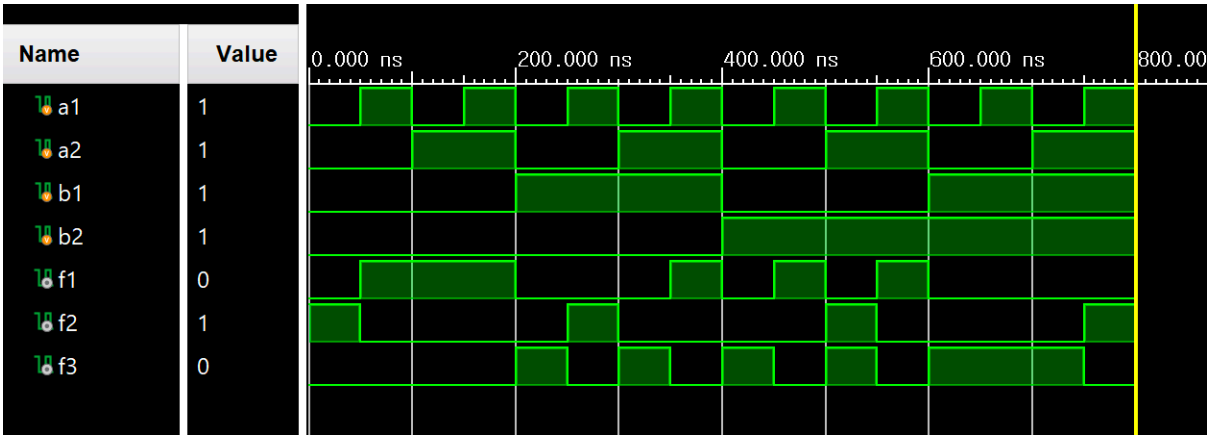
assign f1 = (a1&(~b1)) | (a2&(~b1)&(~b2)) | (a1&a2&(~b2)),

assign f2 = (~(a1^b1)) & (~(a2^b2)),

assign f3 = ((~a1)&b1) | ((~a1)&(~b1)&b2) | ((~a2)&b1&b2)와 같이 나타낼 수 있다. (간소화 방법은 후술) Schematic과 Simulation 결과는 다음과 같다.

```
1   `timescale 1ns / 1ps
2   module two_bit_cmp(
3       input a1, a2, b1, b2,
4       output f1, f2, f3
5       );
6       assign f1 = (a1&(~b1)) | (a2&(~b1)&(~b2)) | (a1&a2&(~b2));
7       assign f2 = (~(a1^b1)) & (~(a2^b2));
8       assign f3 = ((~a1)&b1) | ((~a1)&(~b1)&b2) | ((~a2)&b1&b2);
9
10  endmodule
11
```





A>B, A=B, A<B 여부를 판단하는 2-bit 이진 비교기이다. 1비트 이진 비교기는 앞선 주차에서 이미 다룬 바 있다.

먼저 각 입력값에 대한 출력값을 아래의 진리표로 정리하였다.

($\mathrm{F}_1: A > B,\ \mathrm{F}_2: A = B,\ \mathrm{F}_3: A < B$)

a	b	c	d	F_1	F_2	F_3
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

↓ 카르노 맵

F_1

CD\AB	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

F_2

CD/AB	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

F_3

CD\AB	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

위 카르노 맵을 정리하면,

F_1 = AC'+BC'D'+ABD'

F_2 = A'B'C'D'+A'BC'D+AB'CD'+ABCD = $\overline{A \oplus C} \cdot \overline{B \oplus D}$

F_3 = A'C+A'B'D+B'CD 로 나타낼 수 있다.

∴ assign f1 = (a1&(~b1)) | (a2&(~b1)&(~b2)) | (a1&a2&(~b2)),

assign f2 = (~(a1^b1)) & (~(a2^b2)),

assign f3 = ((~a1)&b1) | ((~a1)&(~b1)&b2) | ((~a2)&b1&b2)와 같이 나타낼 수 있다.

4. 결과 검토 및 논의 사항.

실험을 통해 Even Parity bit generator와 Even Parity bit checker, 2-bit binary comparator의 작동을 확인할 수 있었다.

4, 5 변수 등의 카르노 맵을 이용해 논리식을 간소화해 이를 Verilog로 코딩하였다. 코딩한 결과를 Simulation으로 확인하였고 Schematic으로 회로도의 모습도 확인하였다.

Even Parity bit generator에서는 1의 개수가 짝수가 필요하므로 입력 비트 a, b, c, d 중 1의 개수가 홀수일 때 1을 출력해야 했다. 이에 XOR gate가 사용되었다. Even Parity bit checker에서는 입력 bit a, b, c, d와 parity bit 중 1의 개수가 홀수 개이면 오류를 확인하는 방식이므로 마찬가지로 a ~ d, parity bit을 XOR 처리하였다.

또한 실습 전 미리 진리표를 채워보며 Adder와 Subtractor의 작동을 예상하였는데, 실험 결과 예측과 동일하게 나왔다. Even Parity bit checker 간소화를 통해서는 5변수 카르노 맵을 다루는 방법을 확실히 익힐 수 있었다.

5. 추가 이론 조사 및 작성.

4-bit binary comparator를 만들고자 한다면 4^4 = 256개의 진리표 상의 행이 필요하다. 이는 공간도 많이 차지하고 비효율적이므로 두 4비트 숫자의 각 비트를 비교하고 그 비교와 가중치를 기반으로 진리표를 작성해 보았다.

A3B3	A2B2	A1B1	A0B0	A>B	A<B	A=B
A3>B3	x	x	x	1	0	0
A3<B3	x	x	x	0	1	0
A3=B3	A2>B2	x	x	1	0	0
A3=B3	A2<B2	x	x	0	1	0
A3=B3	A2=B2	A1>B1	x	1	0	0
A3=B3	A2=B2	A1<B1	x	0	1	0

A3=B3	A2=B2	A1=B1	A0>B0	1	0	0
A3=B3	A2=B2	A1=B1	A0<B0	0	1	0
A3=B3	A2=B2	A1=B1	A0=B0	0	0	1

세 가지 출력에 대한 방정식을 유도해 보자면,

1) A = B

위의 진리표를 본다면, A = B를 확인하기 위한 연산은 A⊕B이다. 이를 x라 부른다면, 전체 4비트 A = B는 x0 * x1 * x2 * x3이다.

2) A > B

A3 * B3′ + x3 * A2 * B2′ + x3 * x2 * A1 * B1′ + x3 * x2 * x1 * A0 * B0′으로 나타낼 수 있다.

3) A < B

A3’ * B3 + x3 * A2’B2 + x3 * x2 * A1’ * B1 + x3 * x2 * x1 * A0’ * B0으로 나타낼 수 있다.

위의 세 방정식을 기반으로 회로를 설계하면, 다음과 같은 4-bit comparator 의 회로도를 얻을 수 있다.