7주차 결과보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20191629 이름: 이주헌

1. Even parity bit generator와 checker simulation 결과 및 FPGA 구현 과정에 대해서 설명하시오.

Even parity bit generator는 검증하려는 데이터 니블을 입력받아 올바른 even parity bit를 생성하는 모듈이다. 즉, 입력받은 데이터 니블 안의 1의 개수가 짝수이면 0, 홀수이면 1을 출력해야 한다. Even parity bit generator의 진리표는 다음과 같다.

Input A	Input B	Input C	Input D	Output P
F	F	F	F	F
F	F	F	T	T
F	F	T	F	T
F	F	T	T	F
F	T	F	F	T
F	T	F	T	F
F	T	T	F	F
F	T	T	T	T
T	F	F	F	T
T	F	F	T	F
T	F	T	F	F
T	F	T	T	T
T	T	F	F	F
T	T	F	T	T
T	T	T	F	T
T	T	T	T	F

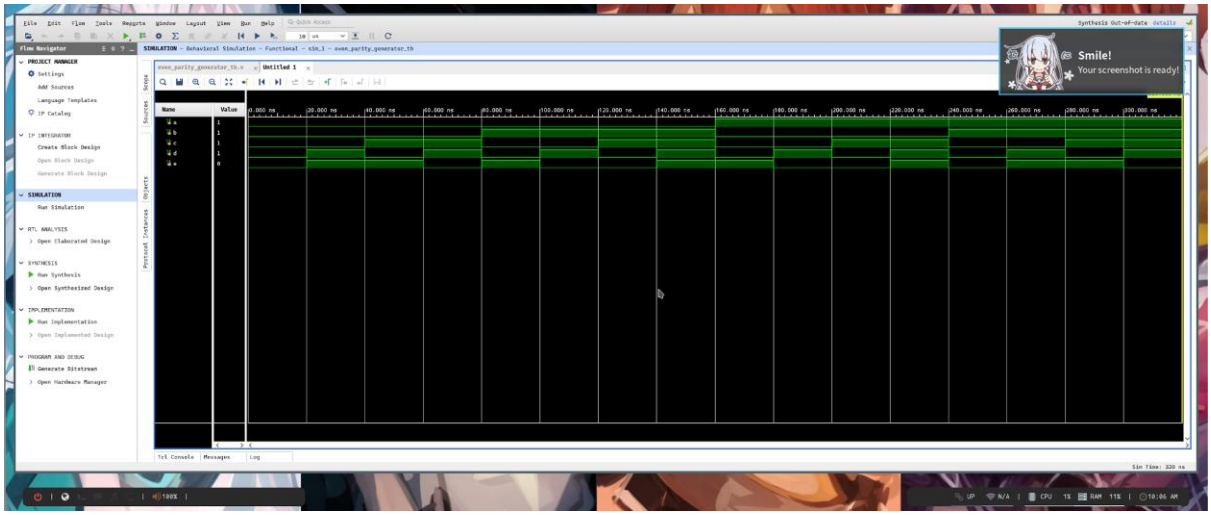
이 진리표를 바탕으로 K-map을 그리면 다음과 같다.

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

위 K-map은 크기가 1인 prime implicant만이 여러 개 존재하는 것처럼 보인다. 이를 실제로 최소화하려면 많은 단계를 거쳐야 하겠지만, 위와 같은 바둑판 모양의 K-map은 XOR 게이트를 사용하여 최소화할 수 있다는 사실이 알려져 있다. 따라서, 출력 P는 다음과 같은 부울 식으로 나타낼 수 있다.

$$P = A \oplus B \oplus C \oplus D$$

이 부울 식을 Verilog로 구현한 뒤 시뮬레이션하면 다음과 같은 결과를 볼 수 있는데, 사진에서 볼 수 있듯이 진리표의 결과와 일치한다.



Even parity checker는 반대로 니블 입력과 parity bit를 입력 받아서 데이터의 안정성을 검증하는 모듈이다. 따라서, 5개의 입력 비트 중 1의 개수가 짝수이면 오류가 발생하지 않았다고 판단하여 0, 홀수이면 오류가 발생했다고 판단하고 1을 출력한다.

Input A	Input B	Input C	Input D	Input P	Result
F	F	F	F	F	F
F	F	F	F	T	T
F	F	F	T	F	T
F	F	F	T	T	F
F	F	T	F	F	T
F	F	T	F	T	F
F	F	T	T	F	F
F	F	T	T	T	T
F	T	F	F	F	T
F	T	F	F	T	F
F	T	F	T	F	F
F	T	F	T	T	T
F	T	T	F	F	F
F	T	T	F	T	T
F	T	T	T	F	T
F	T	T	T	T	F
T	F	F	F	F	T
T	F	F	F	T	F
T	F	F	T	F	F
T	F	F	T	T	T
T	F	T	F	F	F
T	F	T	F	T	T
T	F	T	T	F	T
T	F	T	T	T	F
T	T	F	F	F	F

T	T	F	F	T	T
T	T	F	T	F	T
T	T	F	T	T	F
T	T	T	F	F	T
T	T	T	F	T	F
T	T	T	T	F	F
T	T	T	T	T	T

이를 바탕으로 K-map을 그리면 다음과 같다. 이 회로는 5개의 입력을 가지고 있기 때문에 2개의 입력과 3개의 입력으로 나눌 수 있었으나 입력 P는 parity bit를 입력받는 것으로 2개의 2×2 K-map으로 나누는 것이 직관적이라고 판단하였다.

P = 0		AB			
		00	01	11	10
CD	00	0	1	0	1
	01	1	0	1	0
	11	0	1	0	1
	10	1	0	1	0

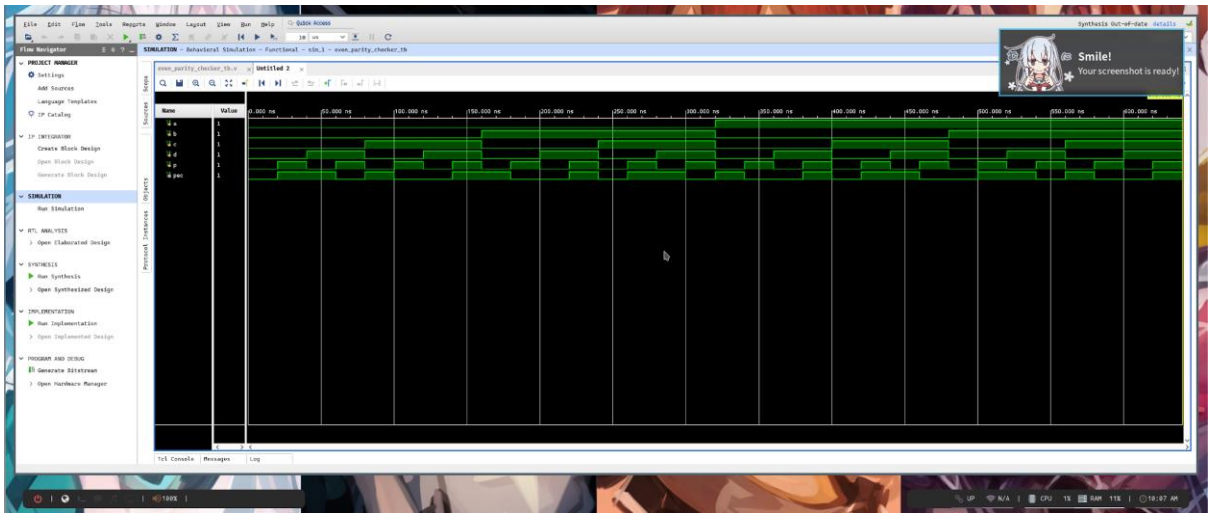
P = 1		AB			
		00	01	11	10
CD	00	1	0	1	0
	01	0	1	0	1
	11	1	0	1	0

	10	0	1	0	1

생성기의 K-map을 최소화할 때와 같은 이유로, 위 K-map을 최소화하면 다음과 같은 부울 식을 구할 수 있다.

$$Result = A \oplus B \oplus C \oplus D \oplus P$$

이 부울 식을 Verilog로 구현한 뒤 시뮬레이션하면 다음과 같은 결과를 볼 수 있는데, 사진에서 볼 수 있듯이 진리표의 결과와 일치한다.



2. Odd parity bit generator와 checker simulation 결과 및 FPGA 구현 과정에 대해서 설명하시오.

반대로, odd parity bit generator는 검증하려는 데이터 니블을 입력받아 올바른 odd parity bit를 생성하는 모듈이다. 즉, 입력받은 데이터 니블 안의 1의 개수가 홀수이면 0, 짝수이면 1을 출력해야 한다. Odd parity bit generator의 진리표는 다음과 같다.

Input A	Input B	Input C	Input D	Output P
F	F	F	F	T
F	F	F	T	F
F	F	T	F	F
F	F	T	T	T
F	T	F	F	F

F	T	F	T	T
F	T	T	F	T
F	T	T	T	F
T	F	F	F	F
T	F	F	T	T
T	F	T	F	T
T	F	T	T	F
T	T	F	F	T
T	T	F	T	F
T	T	T	F	F
T	T	T	T	T

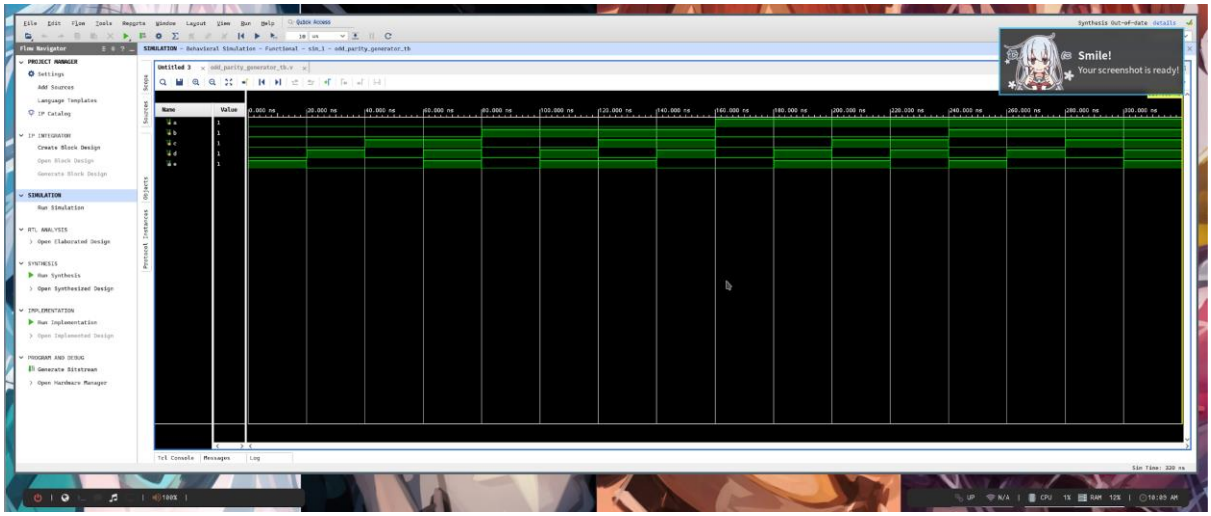
이 진리표를 바탕으로 K-map을 그리면 다음과 같다.

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

위 K-map을 1번 항목의 K-map과 비교해 보았을 때, 출력 결과가 정 반대인 것을 알 수 있다. 위 K-map을 최소화하면 1번 항목의 최소화 결과를 반전시킨 것과 같게 된다는 사실을 쉽게 알 수 있다. 이를 부울 식으로 표현하면 다음과 같다.

$$P = \neg(A \oplus B \oplus C \oplus D)$$

이 부울 식을 Verilog로 구현한 뒤 시뮬레이션하면 다음과 같은 결과를 볼 수 있는데, 사진에서 볼 수 있듯이 진리표의 결과와 일치한다.



Odd parity checker는 니블 입력과 parity bit를 입력 받아서 데이터의 안정성을 검증하는 모듈이다. 따라서, 5개의 입력 비트 중 1의 개수가 홀수이면 오류가 발생하지 않았다고 판단하여 0, 짝수이면 오류가 발생했다고 판단하고 1을 출력한다.

Input A	Input B	Input C	Input D	Input P	Result
F	F	F	F	F	T
F	F	F	F	T	F
F	F	F	T	F	F
F	F	F	T	T	T
F	F	T	F	F	F
F	F	T	F	T	T
F	F	T	T	F	T
F	F	T	T	T	F
F	T	F	F	F	F
F	T	F	F	T	T
F	T	F	T	F	T
F	T	F	T	T	F
F	T	T	F	F	T
F	T	T	F	T	F
F	T	T	T	F	F

F	T	T	T	T	T
T	F	F	F	F	F
T	F	F	F	T	T
T	F	F	T	F	T
T	F	F	T	T	F
T	F	T	F	F	T
T	F	T	F	T	F
T	F	T	T	F	F
T	F	T	T	T	T
T	T	F	F	F	T
T	T	F	F	T	F
T	T	F	T	F	F
T	T	F	T	T	T
T	T	T	F	F	F
T	T	T	F	T	T
T	T	T	T	F	T
T	T	T	T	T	F

이를 바탕으로 K-map을 그리면 다음과 같다. 1번 항목에서 서술한 이유와 마찬가지인 이유로, 위 진리표를 2개의 2×2 K-map으로 표현하였다.

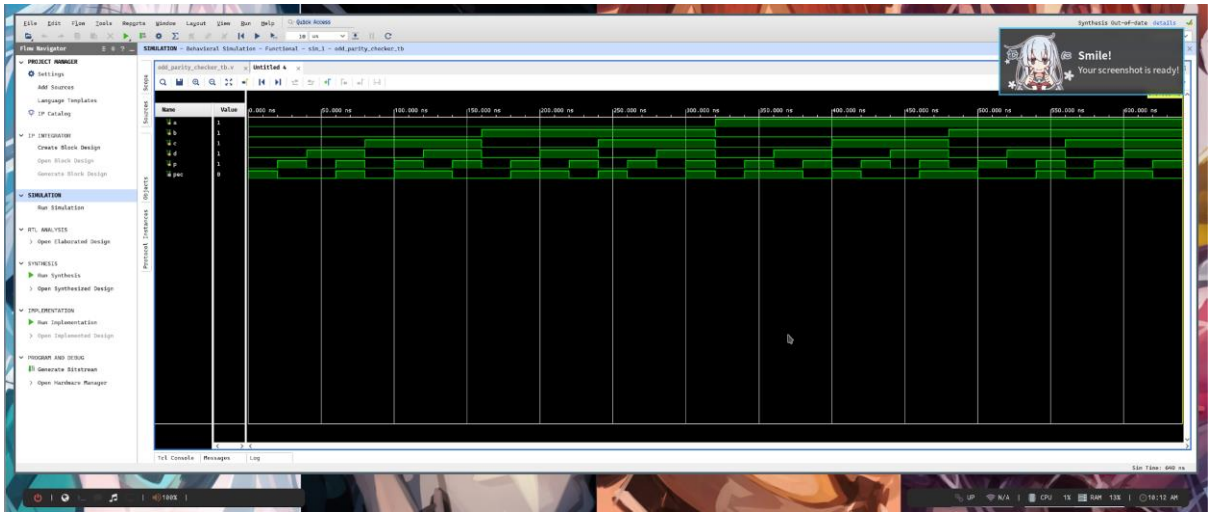
P = 0		AB			
		00	01	11	10
CD	00	1	0	1	0
	01	0	1	0	1
	11	1	0	1	0
	10	0	1	0	1

P = 1		AB			
		00	01	11	10
CD	00	0	1	0	1
	01	1	0	1	0
	11	0	1	0	1
	10	1	0	1	0

생성기의 K-map을 최소화할 때와 마찬가지로, 위 K-map은 1번 항목의 K-map과 완전히 반대되는 모양을 가진다. 따라서, 위 K-map은 다음과 같이 최소화할 수 있다.

$$Result = \neg(A \oplus B \oplus C \oplus D \oplus P)$$

이 부울 식을 Verilog로 구현한 뒤 시뮬레이션하면 다음과 같은 결과를 볼 수 있는데, 사진에서 볼 수 있듯이 진리표의 결과와 일치한다.



3. 2-bit binary comparator의 simulation 결과 및 FPGA 구현 과정에 대해서 설명하시오.

두 개의 숫자 A, B를 비교할 때 생각할 수 있는 경우의 수는 3가지이다.

- A가 B보다 크거나,
- A가 B보다 작거나,

- A와 B가 같다.

만약 두 수가 이진수로 입력된다면 다음과 같이 진리표를 작성할 수 있다.

Input A	Input B	Input C	Input D	AB > CD	AB = CD	AB < CD
F	F	F	F	F	T	F
F	F	F	T	F	F	T
F	F	T	F	F	F	T
F	F	T	T	F	F	T
F	T	F	F	T	F	F
F	T	F	T	F	T	F
F	T	T	F	F	F	T
F	T	T	T	F	F	T
T	F	F	F	T	F	F
T	F	F	T	T	F	F
T	F	T	F	F	T	F
T	F	T	T	F	F	T
T	T	F	F	T	F	F
T	T	F	T	T	F	F
T	T	T	F	T	F	F
T	T	T	T	F	T	F

각 출력 결과에 대해 K-map을 그리면 다음과 같이 3개의 작은 K-map으로 나타낼 수 있다.

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

	11	0	0	0	0
	10	0	0	1	0

AB > CD

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

AB = CD

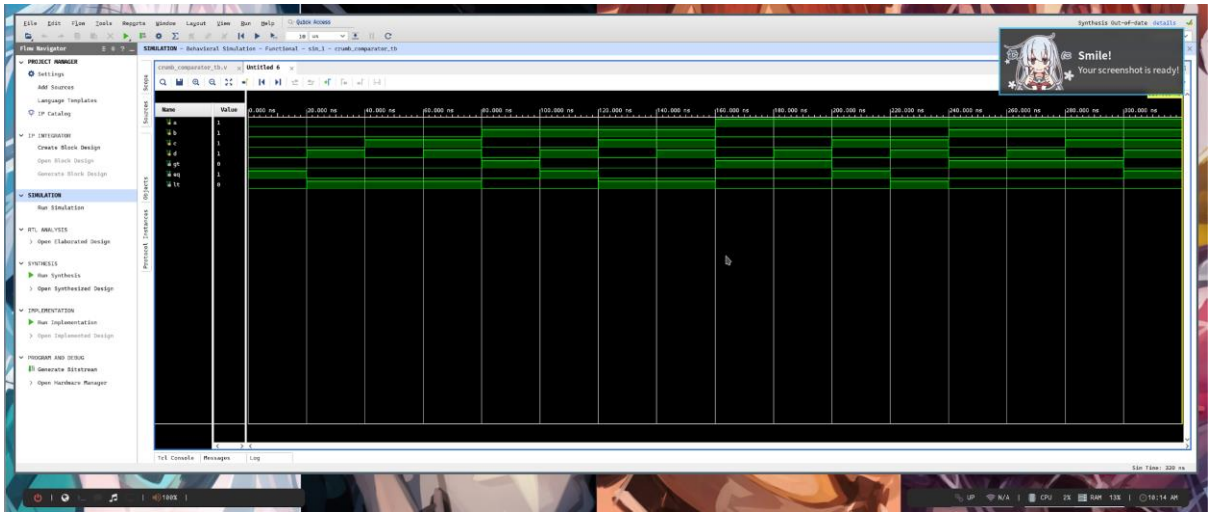
		AB			
		00	01	11	10
CD	00	0	0	0	0
	01	1	0	0	0
	11	1	1	0	1
	10	1	1	0	0

AB < CD

위 K-map을 최소화하면 다음과 같은 부울 식 3개를 얻을 수 있다.

- $AB > CD \Rightarrow A\neg C + B\neg C\neg D + AB\neg D$
- $AB = CD \Rightarrow \neg A\neg B\neg C\neg D + \neg AB\neg CD + A\neg BC\neg D + ABCD = \neg(A \oplus C)\neg(B \oplus D)$
- $AB > CD \Rightarrow \neg AC + \neg A\neg BD + B\neg CD$

이 부울 식을 Verilog로 구현하여 시뮬레이션하면 다음과 같은 결과를 볼 수 있다.



4. 결과 검토 및 논의사항

첫 번째와 두 번째 실습으로 먼저 parity generator와 checker를 구현하였다. Even parity와 odd parity 모두 XOR 게이트를 사용해서 구성할 수 있다는 점을 알 수 있었는데, 이는 입 중 1의 개수가 짝수인지 홀수인지를 판별하는 가장 간단한 게이트이기 때문이다. 다만 XOR 게이트는 1의 개수가 홀수 개일 때에만 논리적 참을 반환하므로 odd parity에서 사용하기 위해서는 NOT 인버터를 사용해 XOR 게이트의 결과를 반전시켜 줘야 한다는 점이 특징이다.

세 번째 실습에서는 2-bit(crumb) 비교기를 만들었는데, 진리표를 바탕으로 K-map을 사용해서 최소화하여 Verilog로 구현하였다. 이전에 구현한 1비트 비교기와 같은 경우는 4개의 출력 (A>B, A=B, A≠B, A<B)을 가지도록 설계했으나 "같지 않다"는 기본적으로 "같다"의 출력에 NOT 인버터를 추가한 것과 같은 결과를 내므로 생략하였다.

5. 추가 이론 조사 및 작성

모든 비교기의 특징으로는, 모든 입력에 대하여 3개 중 단 하나의 출력만 논리적 참이고, 나머지는 논리적 거짓이라는 것이다. 이 특징을 이용하면, 굳이 모든 출력에 대해 진리표를 그리고 최소화 작업을 진행하지 않고도 답을 구할 수 있다.

각 출력에 대해 나머지 두 출력만을 이용하는 논리는 다음과 같이 작성할 수 있다.

- 만약 A가 B보다 크지 않고, A와 B가 같지 않다면 → A는 B보다 작다.
- 만약 A가 B보다 크지 않고, A가 B보다 작지 않다면 → A는 B와 같다.
- 만약 A와 B가 같지 않고, A가 B보다 작지 않다면 → A는 B보다 크다.

즉, 출력 X, Y, Z가 있을 때, 다음이 모두 성립한다.

- $X = \neg Y \neg Z = \neg(Y + Z)$
- $Y = \neg X \neg Z = \neg(X + Z)$
- $Z = \neg X \neg Y = \neg(X + Y)$

따라서, 비교기의 경우 하나의 출력은 다른 두 출력의 논리합의 역을 취하여 더 쉽게 구할 수 있다는 점을 알 수 있었다.