8주차 결과보고서

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

1. 실험 목적

7-segment의 동작 원리를 이해하고 Verilog를 사용하여 7-segment driver를 구현할 수 있다.

2. 7-segment display의 결과 및 simulation 과정에 대해서 설명하시오.

7-segment display를 사용하는 방법은 크게 두 가지가 있는데, 하나는 각 핀에 대한 논리 식을 구하는 것이고 두 번째 방식은 메모리에 우리가 원하는 식을 미리 집어넣고 데이터 버스를 거쳐 가져오는 방식이다. 여기서는 첫 번째 방식으로 7-segment display driver를 구현하였다.

그런데 드라이버를 만들기 전, 각 핀에 대한 논리식을 구해야 한다. 각 숫자에 대하여 원하는 출력값을 정하고, 이를 진리표로 만들면 다음과 같다.

A	B	C	D	A	B	C	D	E	F	G	DP
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1	0
0	0	1	1	1	1	1	1	0	0	1	0
0	1	0	0	0	1	1	0	0	1	1	0
0	1	0	1	1	0	1	1	0	1	1	0
0	1	1	0	1	0	1	1	1	1	1	0
0	1	1	1	1	1	1	0	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1	0
1	0	0	1	1	1	1	0	0	1	1	0
1	0	1	0	1	1	1	0	1	1	1	0

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

여러 개의 출력이 있으므로 각 출력에 대한 논리식을 개별적으로 구해 주어야 하는데, 이를 달성하는 가장 좋은 방법은 각 출력에 대한 카르노 맵을 그리는 것이다. DP의 출력은 항상 0이므로 $DP = 0$으로 놓고, 나머지 7개 입력에 대해 카르노 맵을 그리면 다음과 같다.

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

Output $A = \neg AC + BC + \neg B\neg D + \neg ABD + A\neg B\neg C + A\neg C\neg D$

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

Output $B = \neg A\neg B + \neg B\neg C + \neg B\neg D + \neg A\neg C\neg D + \neg ACD + A\neg CD$

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

Output $C = \neg CD + \neg AB + A\neg B + \neg A\neg C + \neg AD$

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

Output $D = \neg B\neg C\neg D + B\neg CD + AB\neg D + \neg BCD + \neg AC\neg D$

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

Output $E = C\neg D + AB + AC + \neg B\neg D$

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

Output $F = A\neg B + \neg C\neg D + AC + \neg AB\neg C + BC\neg D$

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

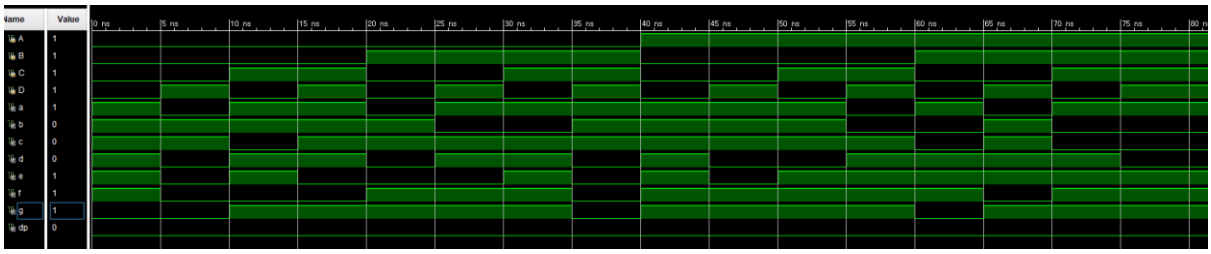
Output $G = A\neg B + C\neg D + AD + \neg BC + \neg AB\neg C$

위에서 구한 결과로부터 다음과 같은 Verilog 코드를 작성할 수 있다.

```
`timescale 1ns / 1ps

module segment(
    input a, b, c, d,
    output A, B, C, D, E, F, G, seg
);

assign A = (b & c) | (!a & c) | (a & !d) | (!b & !d) | (!a & b & d) | (a
& !b & !c);
```

```
assign B = (!b & !d) | (!b & !c) | (!a & c & d) | (a & !c & d) | (!a
& !c & !d);
assign C = (!c & d) | (!a & d) | (a & !b) | (!a & b) | (!b & !c);
assign D = (!a &c & !d) | (!b & c & d) | (b & !c & d) | (a & b & !d) |
(!b & !c & !d);
assign E = (c & !d) | (a & c) | (a & b) | (!b & !d);
assign F = (a & c) | (a & !b) | (b & !d) | (!c & !d) + (!a & b & !c);
assign G = (a & c) | (!b & c) | (a & !b) | (b & !c & d) | (!a & b & !d);
assign seg = 1;

endmodule
```

위 모듈에 대하여 A, B, C, D, 4개 모든 입력을 집어넣어 보면 다음과 같은 시뮬레이션 결과를 얻을 수 있다.



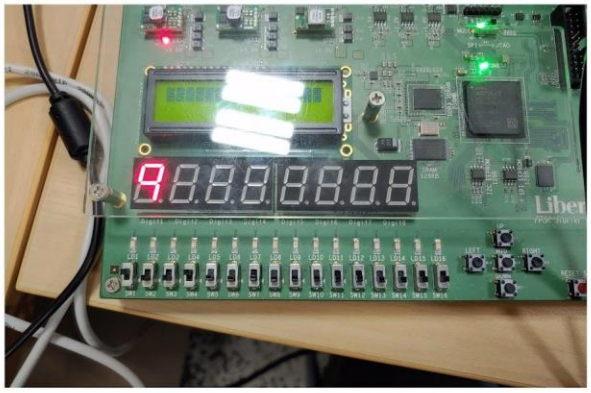
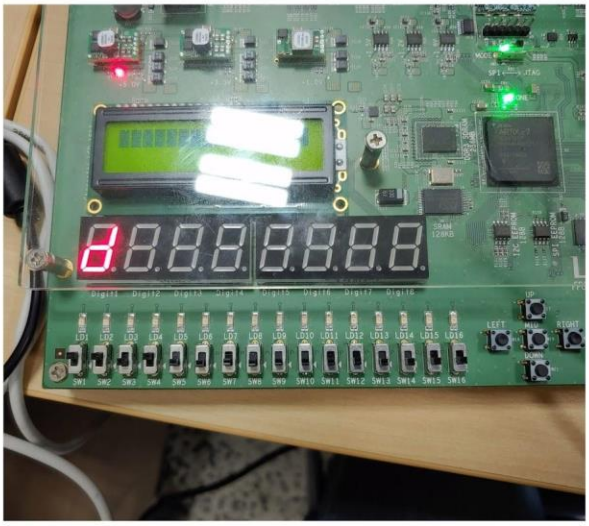
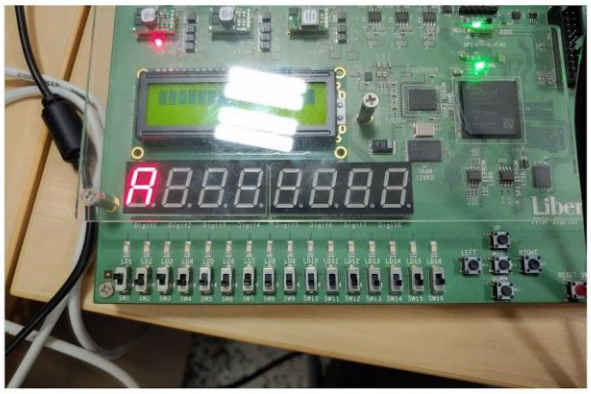
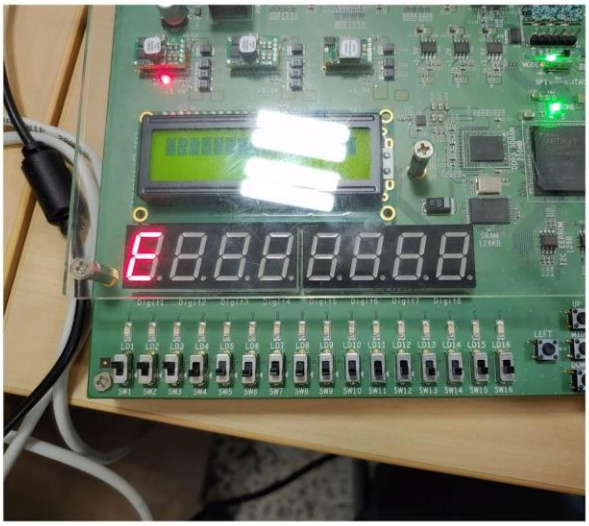
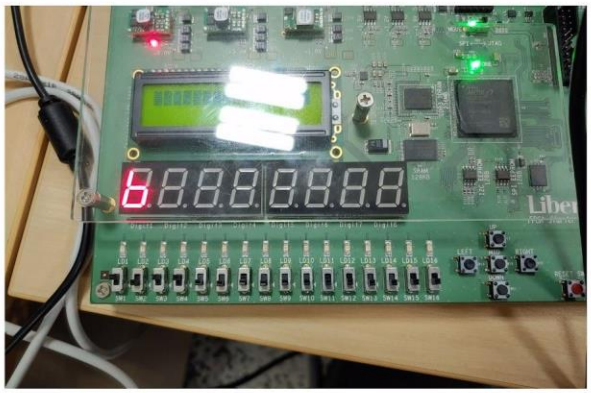
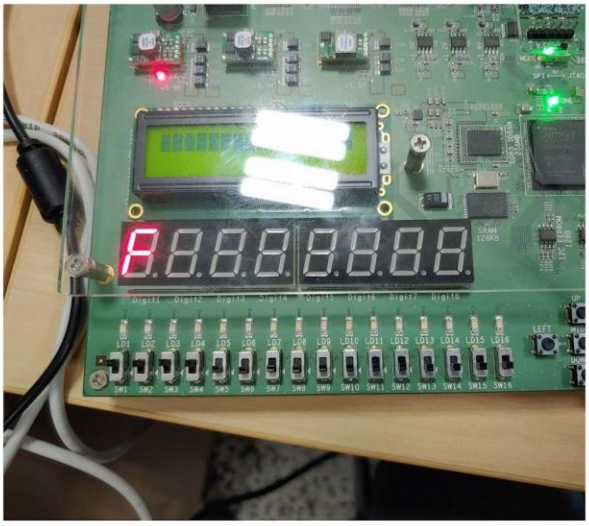
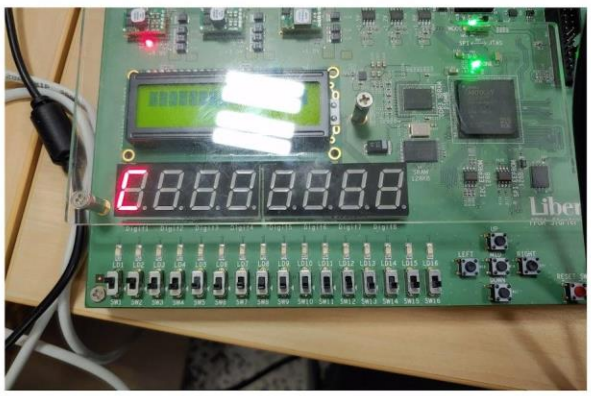
입력값의 결과와 진리표의 결과를 비교해 보면 둘이 서로 같다는 사실을 알 수 있다. 마지막으로, FPGA의 7-segment display로 프로그래밍하면 다음과 같은 결과를 볼 수 있는데, 우리가 원했던 결과와 같다는 것을 알 수 있다.









3. 결과 검토 및 논의 사항

이번 실습에서는 7-segment에 입력한 이진수 숫자를 표시하는 실험을 진행하였다. 이번 실습에서는 특히 출력 핀이 7개나 되어 많은 카르노 맵을 그릴 필요가 있었다. 카르노 맵을 이용하여 논리 식을 작성한 뒤, 각 핀에 맞게 Verilog로 논리회로를 구현해 주었다.

4. 추가 이론 조사 및 작성

7-segment display driver를 만드는 방식은 위에서도 언급했듯이 두 가지이다. 이번에 사용한 방법 이외에 다른 방법은 각 입력에 대한 출력값을 미리 전부 추가하는 것이다. 이는 논리 게이트를 사용하기 않기 때문에 여러 논리 게이트를 거치지 않아 속도가 빠르다는 장점이 있지만, 그만큼 메모리 용량을 많이 사용하게 된다. 이와 같은 방식의 드라이버는 다음과 같이 작성할 수 있다.

```
module segment(
    input  [3:0]x,
    output reg [6:0]z
);
always @*
case (x)
4'b0000:
    z = 7'b1111110;
4'b0001:
    z = 7'b0110000;
4'b0010:
    z = 7'b1101101;
4'b0011:
    z = 7'b1111001;
4'b0100:
    z = 7'b0110011;
4'b0101:
    z = 7'b1011011;
4'b0110:
    z = 7'b1011111;
4'b0111:
    z = 7'b1110000;
4'b1000:
    z = 7'b1111111;
4'b1001:
    z = 7'b1111011;
4'b1010:
    z = 7'b1110111;
4'b1011:
    z = 7'b0011111;
```

```
4'b1100:
    z = 7'b1001110;
4'b1101:
    z = 7'b0111101;
4'b1110:
    z = 7'b1001111;
4'b1111:
    z = 7'b1000111;
endcase

endmodule
```

이 경우는 Verilog의 case 분기문을 이용하는 것이 간편하기 때문에 입력 신호를 a, b, c, d로 나누지 않고 4바이트 입력으로 한꺼번에 입력받았다. 출력 신호 7개의 핀을 개별적으로 설정하는 것보다 하나의 바이트 출력을 나누는 편이 편리하기 때문에 7비트 출력을 받을 수 있도록 설정했다.