8주차 결과보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20191583 이름: 김태곤

**1. 실험 목적**

Verilog를 사용하여 7-segment display를 디자인하고 구현하는 과정을 통해 회로 설계의 기본 원리와 과정을 실습하고 이해한다. 이를 통해 다음과 같은 구체적인 경험과 지식을 얻는다. 7-segment display의 기본 구조와 동작 원리를 학습한다. Segment가 어떻게 구성되어 있으며, 이를 통해 숫자와 일부 문자를 어떻게 표현할 수 있는지 이해한다. 이후 Verilog를 사용하여 7-segment display의 동작을 모델링한다. 숫자와 입력 신호에 따른 세그먼트의 활성화를 제어하는 논리식을 설계한다. 이후 테스트 벤치를 구축하여 입력 신호를 디스플레이 논리 회로에 적용하고, simulation을 통해 각 게이트의 입력과 출력 값을 검증하며 회로의 정확성을 확인한다. 최종적으로 FPGA 보드에 구현하여 실제 하드웨어에서 디스플레이 동작을 검증하고, 설계의 효율성과 정확성을 평가한다.

**2. 7-Segment Display의 결과 및 Simulation 과정에 대해서 설명하시오.**

**(Truth table 작성 및 k-map 포함,0~F,DP)(최대한 상세히 기술할 것)**

**※ 모든 simulation, 진리표, 카르노맵 결과는 직접 simulation/작성하여 얻은 결과로만 보고서 작성하셔야 합니다.**

**강의자료 또는 인터넷 검색으로 얻은 자료 사용 시 감점됩니다.**

1. **Truth Table**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **input** | | | | **Output** | | | | | | | |
| **w** | **x** | **y** | **z** | **a** | **b** | **c** | **d** | **e** | **f** | **G** | **Dp** |
| **0** | **0** | **0** | **0** | 1 | 1 | 1 | 1 | 1 | 1 | 0 | **1** |
| **0** | **0** | **0** | **1** | 0 | 1 | 1 | 0 | 0 | 0 | 0 | **1** |
| **0** | **0** | **1** | **0** | 1 | 1 | 0 | 1 | 1 | 0 | 1 | **1** |
| **0** | **0** | **1** | **1** | 1 | 1 | 1 | 1 | 0 | 0 | 1 | **1** |
| **0** | **1** | **0** | **0** | 0 | 1 | 1 | 0 | 0 | 1 | 1 | **1** |
| **0** | **1** | **0** | **1** | 1 | 0 | 1 | 1 | 0 | 1 | 1 | **1** |
| **0** | **1** | **1** | **0** | 1 | 0 | 1 | 1 | 1 | 1 | 1 | **1** |
| **0** | **1** | **1** | **1** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | **1** |
| **1** | **0** | **0** | **0** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | **1** |
| **1** | **0** | **0** | **1** | 1 | 1 | 1 | 0 | 0 | 1 | 1 | **1** |
| **1** | **0** | **1** | **0** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | **1** |
| **1** | **0** | **1** | **1** | 0 | 0 | 1 | 1 | 1 | 1 | 1 | **1** |
| **1** | **1** | **0** | **0** | 1 | 0 | 0 | 1 | 1 | 1 | 0 | **1** |
| **1** | **1** | **0** | **1** | 0 | 1 | 1 | 1 | 1 | 0 | 1 | **1** |
| **1** | **1** | **1** | **0** | 1 | 0 | 0 | 1 | 1 | 1 | 1 | **1** |
| **0** | **1** | **1** | **1** | 1 | 0 | 0 | 0 | 1 | 1 | 1 | **1** |

1. **K-map**

**K-map : a**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **wx\yz** | **00** | **01** | **11** | **10** |
| **00** | **1** | **0** | **1** | **1** |
| **01** | **0** | **1** | **1** | **1** |
| **11** | **1** | **0** | **1** | **1** |
| **10** | **1** | **1** | **0** | **1** |

**a = wx’y’+w’xz+wz’+w’y+xy+x’z’**

**K-map : c**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **wx\yz** | **00** | **01** | **11** | **10** |
| **00** | **1** | **1** | **1** | **0** |
| **01** | **1** | **1** | **1** | **1** |
| **11** | **0** | **1** | **0** | **0** |
| **10** | **1** | **1** | **1** | **1** |

**c = w’y’+w’z+y’z+w’x+wx’**

**K-map : e**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **wx\yz** | **00** | **01** | **11** | **10** |
| **00** | **1** | **0** | **0** | **1** |
| **01** | **0** | **0** | **0** | **1** |
| **11** | **1** | **1** | **1** | **1** |
| **10** | **1** | **0** | **1** | **1** |

**e = x’z’+yz’+wx+wy**

**K-map : b**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **wx\yz** | **00** | **01** | **11** | **10** |
| **00** | **1** | **1** | **1** | **1** |
| **01** | **1** | **0** | **1** | **0** |
| **11** | **0** | **1** | **0** | **0** |
| **10** | **1** | **1** | **0** | **1** |

**b = w’y’z’+w’yz+wy’z+w’x’+x’z’**

**K-map : d**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **wx\yz** | **00** | **01** | **11** | **10** |
| **00** | **1** | **0** | **1** | **1** |
| **01** | **0** | **1** | **0** | **1** |
| **11** | **1** | **1** | **0** | **1** |
| **10** | **1** | **0** | **1** | **0** |

**d = w’x’z’+x’yz+wy’z+wyz’+wy’z’**

**K-map : f**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **wx\yz** | **00** | **01** | **11** | **10** |
| **00** | **1** | **0** | **0** | **0** |
| **01** | **1** | **1** | **0** | **1** |
| **11** | **1** | **0** | **1** | **1** |
| **10** | **1** | **1** | **1** | **1** |

**f = w’xy’+y’z’+xz’+wy+wx’**

**K-map : g**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **wx\yz** | **00** | **01** | **11** | **10** |
| **00** | **0** | **0** | **1** | **1** |
| **01** | **1** | **1** | **0** | **1** |
| **11** | **0** | **1** | **1** | **1** |
| **10** | **1** | **1** | **1** | **1** |

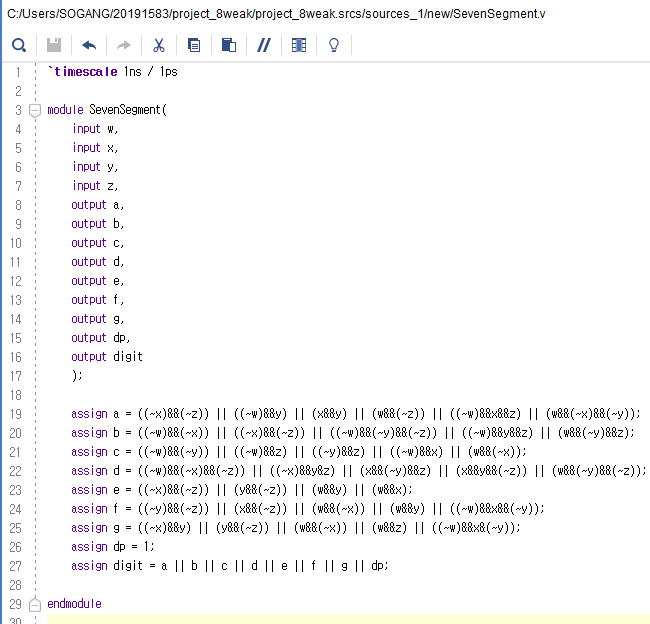
**g = x’y+yz’+wx’+wz+w’xy’**

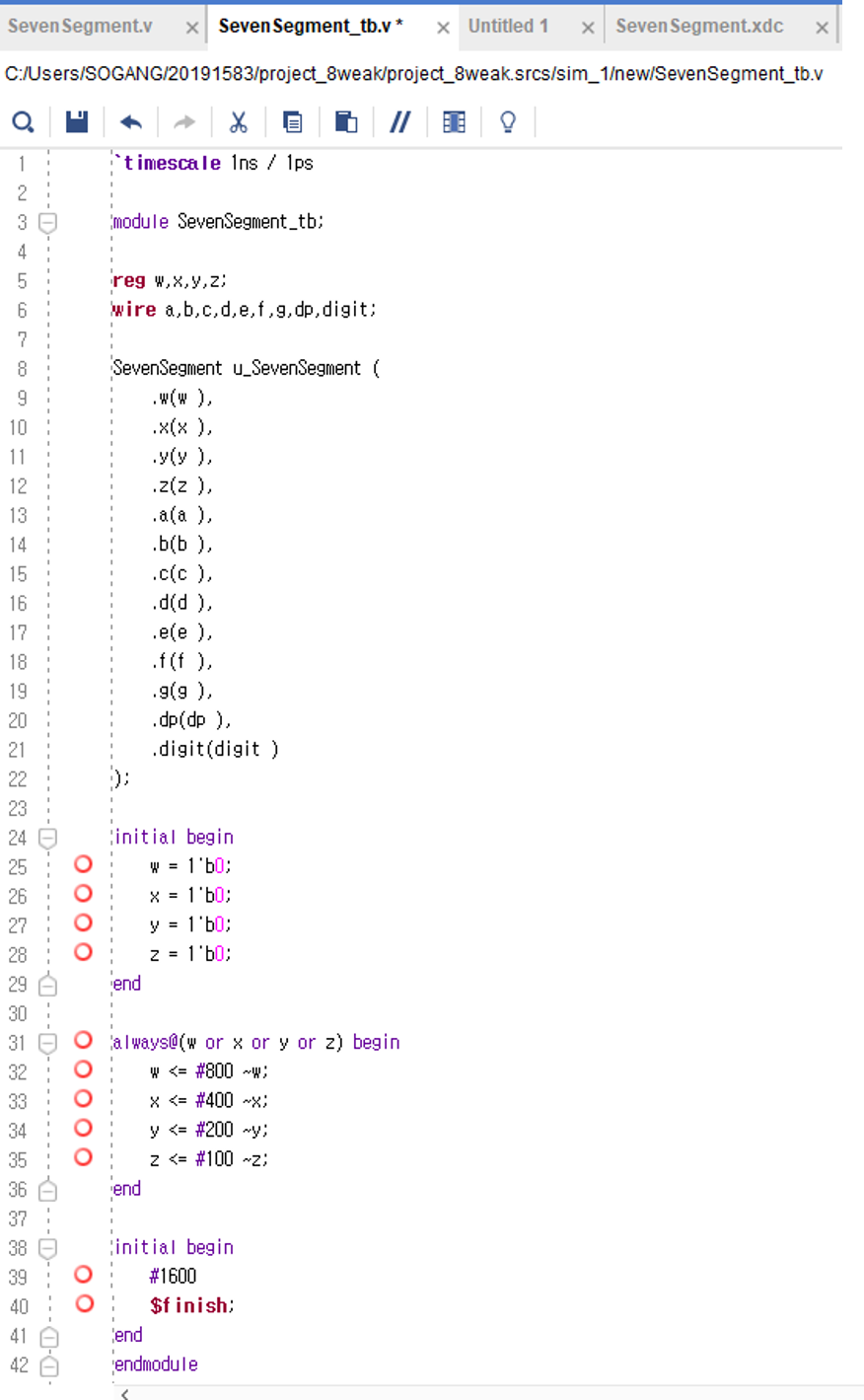
**K-map : dp**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **wx\yz** | **00** | **01** | **11** | **10** |
| **00** | **1** | **1** | **1** | **1** |
| **01** | **1** | **1** | **1** | **1** |
| **11** | **1** | **1** | **1** | **1** |
| **10** | **1** | **1** | **1** | **1** |

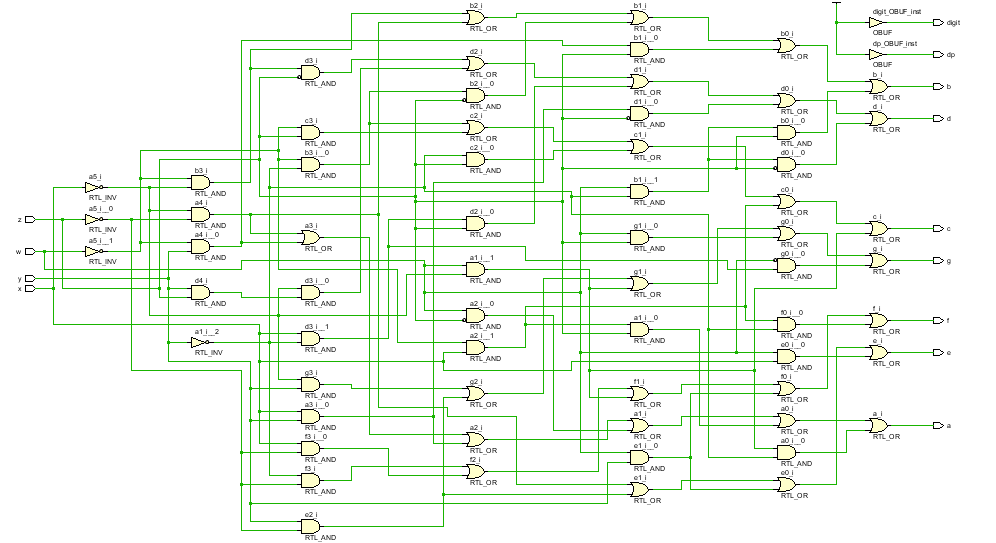
**dp = 1**

1. **Design Source & Test Bench Code**

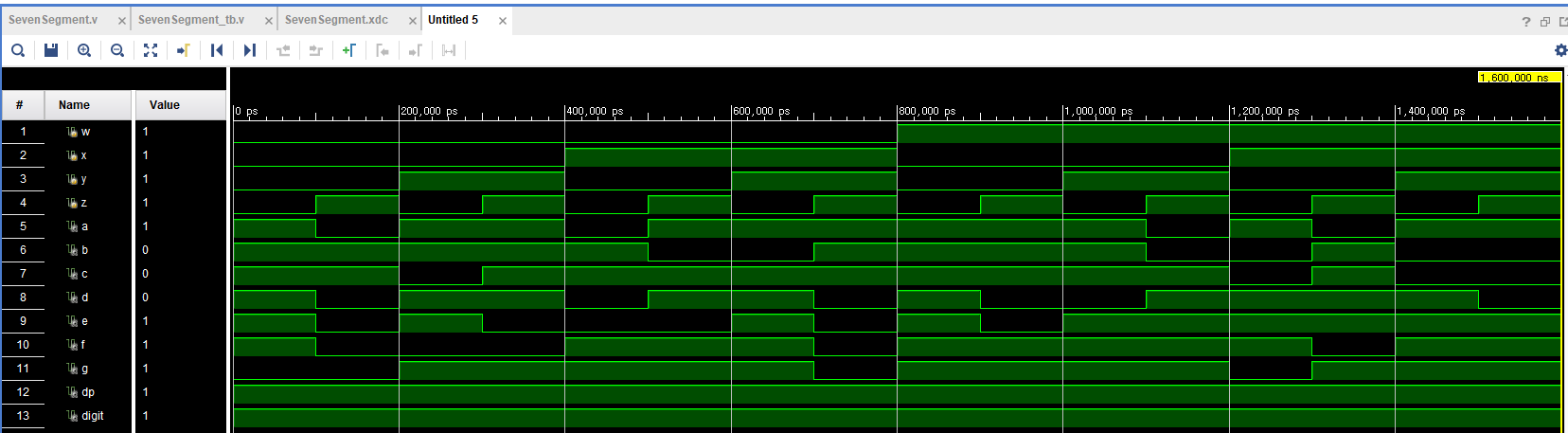
****

****

1. **Schematic**

****

1. **Simulation**

****

1. **결과 및 과정**

7-segment display를 구현하기 위해 각 segment에 대한 논리식은 다음과 같이 작성하였다.

assign a = wx’y’ + w’xz + wz’ + w’y + xy + x’z’;

assign b = w’y’z’ + w’yz + wy’z + w’x’ + x’z’;

assign c = w’y’ + w’z + y’z + w’x + wx’;

assign d = w’x’z’ + x’yz + wy’z + wyz’ + wy’z’;

assign e = x’z’ + yz’ + wx + wy;

assign f = w’xy’ + y’z’ + xz’ + wy + wx’;

assign g = x’y + yz’ + wx’ + wz + w’xy’;

assign dp = 1;

dp의 경우 1을 지정하여 항상 점등되게 지정하였다.

Testbench에서는 w,x,y,z 신호가 순차적으로 변하도록 각각 800,400,200,100 마다 전환되게 하였다.

추가적으로, FPGA에서 특정 위치에 있는 7-segment display에 출력을 하기 위해서는 해당 디스플레이에 연결된 DIGIT 핀에 전압(HIGH=1)을 인가해야 한다. 따라서 assign digit=a||b||c||d||e||f||g||dp 를 통해 1을 부여하였고, E14 pin에 할당하였다.

시뮬레이션 결과, 0에서 15까지의 입력 값에 대해 7-세그먼트 디스플레이에 0부터 9까지의 숫자와 A, b, C, d, E, F가 정확하게 표시되는 것을 확인하였다. 또한 Verilog 코드를 FPGA에 다운로드하여 하드웨어에서 동작을 검증한 결과, 각 입력 값에 대해 7-segment display에 정확한 값이 나타났다. 이로써 Verilog 언어로 설계한 7-Segment display가 예상한 진리표대로 올바르게 작동함을 확인할 수 있었고, FPGA 보드와 7-segment display를 이용해 0부터 15까지의 숫자 및 문자를 성공적으로 표시할 수 있었다.

**3. 결과 검토 및 논의 사항.**

이번 실험을 통해 Verilog로 설계한 7-segment display가 예상한 대로 정확하게 작동함을 확인할 수 있었다. 시뮬레이션과 FPGA 보드에서 다음과 같은 결과를 얻을 수 있었다. 시뮬레이션 결과 Verilog 코드에서 각 세그먼트의 논리식을 정확하게 구현하여 시뮬레이션 결과 예상한 진리표에 부합하는 결과를 얻었다. 또한 0에서 15까지의 입력 값에 대해 7-세그먼트 디스플레이에 0부터 9까지의 숫자와 A, b, C, d, E, F가 정확하게 표시되는 것을 확인하였다. FPGA를 통해서는 각 입력 값에 대해 7-세그먼트 디스플레이에 0부터 9까지의 숫자와 A, b, C, d, E, F가 정확하게 표시되는 것을 확인할 수 있었다.

Verilog 언어로 설계된 7-segment display는 예상된 진리표대로 정확하게 동작하였고, FPGA에서 하드웨어적으로 성공적으로 구현되었다. 시뮬레이션 및 FPGA 테스트를 통해 모든 숫자 및 문자가 정확하게 출력되는 것을 확인함으로써 논리식과 회로 설계의 정확성을 입증할 수 있었다.

추가로 현재의 디자인은 4비트 입력을 통해 16진수 숫자와 문자를 표현하지만, 추가적인 기능을 통해 더 복잡한 출력 패턴이나 알파벳 전체를 표현할 수 있다. 예를 들어, 다중 7-세그먼트 디스플레이를 제어하거나 특수 기호를 표시하는 기능을 추가할 수 있다. 추후 보다 다양한 segment display를 구현하면 학습에 도움이 될 것 같다.

**4. 추가 이론 조사 및 작성.**

7-segment display를 통해 0~9와 a~f(10~15)를 나타내었다. 추가로 7-segment display는 숫자 뿐만 아니라 다양한 형태의 알파벳도 표현할 수 있다.

7-세그먼트 디스플레이를 사용하여 알파벳 문자를 나타내려면 각 알파벳 문자에 대한 적절한 세그먼트를 점등시켜야 한다. 다음은 각 알파벳 문자를 나타내기 위한 세그먼트 조합이다.

|  |  |  |
| --- | --- | --- |
| 알파벳 | 세그먼트 조합 | 설명 |
| A | a, b, c, e, f, g | 01110111 |
| b | c, d, e, f, g | 01111100 |
| C | a, d, e, f | 00111001 |
| d | b, c, d, e, g | 01011110 |
| E | a, d, e, f, g | 01111001 |
| F | a, e, f, g | 01110001 |
| G | a, c, d, e, f | 00111101 |
| H | b, c, e, f, g | 01110110 |
| L | d, e, f | 00111000 |
| P | a, b, e, f, g | 01110011 |
| r | e, g | 00101000 |
| S | a, c, d, f, g | 01101101 |
| U | b, c, d, e, f | 00111110 |
| y | b, c, d, f, g | 01101110 |
| - | g | 01000000 |

7-segment display는 숫자뿐만 아니라 라틴, 키릴 및 그리스 알파벳의 대부분의 글자와 구두점을 표시하는 데 사용할 수 있다. 특별한 경우로 10부터 15까지의 16진수 값을 나타낼 때 문자 A부터 F까지를 표시하는 것이 있다. 이러한 기능은 과학 계산기나 전자 장비의 테스트 디스플레이에서 사용한다.

7-segment display는 단순한 숫자 표시 이외에도 다양한 기호와 알파벳을 표현할 수 있지만, 특정 알파벳 글자는 모호하거나 명확하게 표현하기 어렵다. 따라서 더 복잡한 메시지 표현을 위해서는 14-세그먼트, 16-세그먼트 또는 도트 매트릭스 디스플레이를 사용하는 것이 일반적이다.

