**기초디지털실험**

**Week1\_Environment\_Setting**

**결과레포트**

****

**학정번호 : EEE3313-07-00**

**학번 : 2018142173**

**이름 : 이 찬**

**작성일자 : 2022-03-13**

**목차**

**1. 실험 목적**

**2. 실험 이론**

* **Test Benchmark Description for simulation**
* **비바도에서 simulation을 돌렸을 때 나오는 결과를 서술하라**
* **Module Define**
* **작성한 Module에 대한 정의**
* **Module Instantiation and Module connection**
* **모듈의 인스턴스화와 모듈 연결을 정의하라**
* **Pin Assignment and Connection to Top Module**

**->**

**3. Waveform simulation results**

**Verilog Circuit Design: addition operation evaluation through simulation**

* **Result analysis**

**4. FPGA results**

**Verilog Circuit Design: show 2-bit addition operation results using LED**

* **Result analysis**

**5. 토의**

**절차**

**1. 모듈 생성하기 위한 design source 생성 ( logic\_gate )**

**2. waveform simulation을 하기 위해 simulation sources 생성 (testbench)**

**3. 성능 측정**

**4. 보드에 나타내기 위해서 constraints 생성 후 보드와 연결할 수 있는 코드 저장**

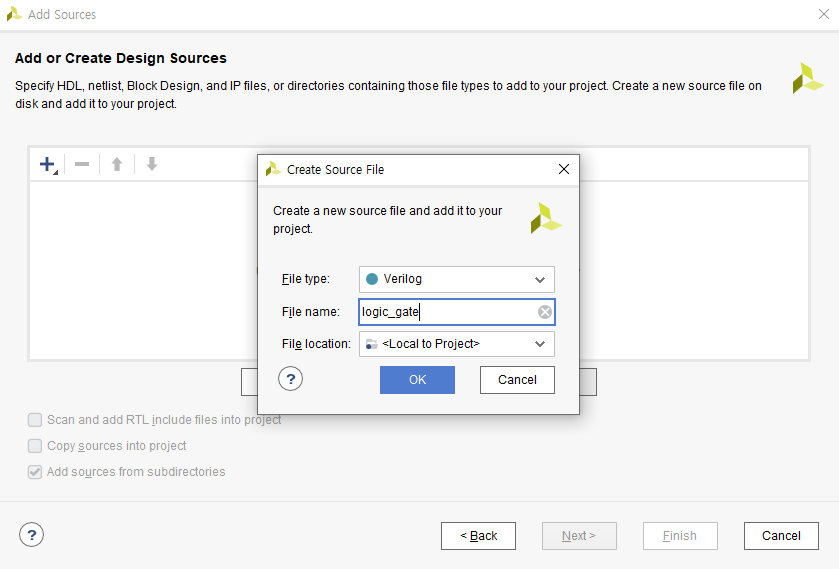
**5. 실제 보드의 핀과 연결하기 위해서 topModule파일 생성**

**6. 보드 연결 후 분석**

텍스트이(가) 표시된 사진

자동 생성된 설명

**새로운 프로젝트를 생성한다.**



**Design sources 생성 및 logic\_gate로 파일 이름 정의**

텍스트이(가) 표시된 사진

자동 생성된 설명

**module logic\_gate ( input [1:0] input\_a, output [3:0] result );**

* **logic\_gate 라는 이름의 Module을 생성한다. input 신호를 생성하기 위해서 input\_a라는 변수를 통해 2-bit signal을 지정한다. output 신호를 생성하기 위해서 result라는 변수를 통해 4-bit 시그널을 지정한다.**

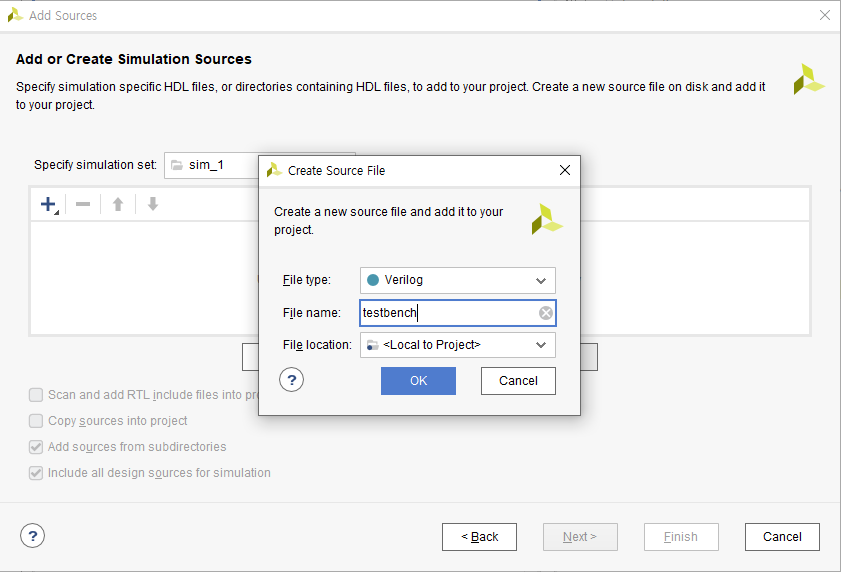
**assign result = input\_a + 3’b111;**

* **module에서 선언한 input\_a 와 3비트의 111의 값을 서로 더한 결과를 result에 할당한다.**

**따라서 input\_a가 가질 수 있는 경우의 수는 00,01,10,11이 되며 각 경우와 3비트의 111값을 더한 값이 result에 나타나게 될 것이다. 이때 처음 module을 설정할 때, result를 4-bit로 설정했으므로 결과값 또한 4비트로 나타날 것임을 예측할 수 있다.**

**결과 표**

|  |  |  |
| --- | --- | --- |
| **Input\_a** | **3’b111** | **Result** |
| **00** | **111** | **0111** |
| **01** | **111** | **1000** |
| **10** | **111** | **1001** |
| **11** | **111** | **1010** |



**생성한 module을 simulation 하기 위해 testbench 파일 생성한다.**

테이블이(가) 표시된 사진

자동 생성된 설명

**module testbench(); : testbench라는 이름의 모듈을 생성한다.**

**reg [1:0]new\_a;**

* **reg type의 변수는 값을 저장할 수 있다. 2-bit signal의 값을 가진 변수 new\_a를 선언한다.**

**wire [3:0]new\_result;**

* **type의 변수는 값을 저장할 수 없다. 4-bit signal의 값을 가진 변수 new\_result을 선언한다.**

**initial begin**

**new\_a <= 2'b11;**

**#10**

**new\_a <= 2'b10;**

**#10**

**new\_a <= 2'b01;**

**#10**

**new\_a <= 2'b00;**

**end**

* **testbench에서 simulation을 작동할 때 한 번만 작동하게 하기 위해서 initial 블록을 사용한다. initial begin이 시작점이고 end는 종료점이다. 변수 new\_a가 가질 수 있는 경우의 수가 총 4가지(00,01,10,11)이므로 각 경우를 시간차에 따라서 new\_a값에 할당한다. 2-bit signal을 이용해서 new\_a가 11부터 00까지 10초 간격으로 new\_a값을 변경하면서 할당하는 블록을 생성하였다.**

**logic\_gate gate\_0( .input\_a(new\_a), .result(new\_result) );**

* **처음에 생성한 logic\_gate 모듈을 인스턴스화하여 모듈에서 정의된 input\_a와 result를 gate\_0이라는 인스턴스로 표현한다. 이때, testbench.v에서 선언한 new\_a와 new\_result을 각각 input\_a, result의 객체로 나타낸다.**

**Waveform Simulation의 결과**

텍스트, 모니터, 스크린샷, 은색이(가) 표시된 사진

자동 생성된 설명

**a값이 변할 때마다 10ns 단위로 r값이 변경되는 것을 확인할 수 있다.**

* **0~10.000ns : a값이 11(3)일 때, r값이 1010**
* **10.000ns~20.000ns : a값이 10(2)일 때, r값이 1001**
* **20.000ns~30.000ns : a값이 01(1)일 때, r값이 1000**
* **30.000ns ~ : a값이 00(0)일 때, r값이 0111 로 변하는 것을 확인할 수 있다.**

**이는 처음에 제시한 결과표와 값이 동일하게 나타나는 것을 알 수 있다.**

텍스트이(가) 표시된 사진

자동 생성된 설명

**Fpga 보드 위에 디지털 회로를 작동시키기 위해서 constraints를 추가한다. 추가하는 파일은 논리 게이트를 작동시킬 때 사용한 파일을 그대로 사용한다. 그 이유는 module의 내용과 상관없이 실제 보드와 각 module을 연결하는 위치는 동일하기 때문이다.**

텍스트이(가) 표시된 사진

자동 생성된 설명

**Design source를 생성하고 topModule로 파일 이름을 정의한다.**

**topModule을 생성하는 이유**

**1. constraints의 각 port를 확인하고 실제 보드와 연결을 하기 위함이다. 2. test benchmark로 simulation한 결과값과 일치하는 지 확인하기 위해서이다.**

텍스트이(가) 표시된 사진

자동 생성된 설명

**Constraints의 코드를 보면, 연결하고자 하는 switches port와 leds port가 각각 정의되어 있는 것을 확인할 수 있다.**

**Switches port의 경우 sw[0]과 sw[1], LEDs port의 경우** **led[0], led[1], led[2], led[3] 로 코드화되어 있는 것을 확인할 수 있다. 따라서 module을 작성할 때 topModule의 객체를 port와 연결되게 작성하면 된다. 작성한 코드는 다음과 같다.**

텍스트이(가) 표시된 사진

자동 생성된 설명

**topModule( input [1:0] sw, output[3:0] led );**

* **constraints에서 정의된 port인 sw와 led를 각각 input, output의 값으로 나타낸다. 이때, sw는 0과 1로 존재하므로 2bit 형태인 [1:0], led는 0,1,2,3으로 존재하므로 4bit 형태인 [3:0]으로 설정한다.**

**wire [1:0]new\_a; wire [3:0]new\_result;**

* **register과 wire의 차이는 값을 저장할 수 있냐/없냐의 차이인데, 작성할 코드에선 굳이 값을 저장할 필요가 없다고 판단하여, input과 output에 대한 변수 new\_a,r 둘 다 wire로 생성하였다.**

**assign new\_a[1] = sw[1]; assign new\_a[0] = sw[0]; assign led[3] = new\_result[3];**

**assign led[2] = new\_result[2]; assign led[1] = new\_result[1]; assign led[0] = new\_result[0];**

* **switch의 port인 sw와 LED의 port인 led를 wire을 통해 선언한 변수 new\_a, new\_result와 각각 대칭시킨다. 이때, switch는 2개, led는 4개가 존재하므로 총 6개의 할당이 진행된다.**

**logic\_gate gate\_0( .input\_a(new\_a), .result(new\_result) );**

* **testbench.v에서 처럼 logic\_gate.v module의 변수를 인스턴스화하여 topModule.v에서 logic\_gate의 input\_a와 result를 각각 new\_a, new\_result로 대칭시켜 객체를 생성한다.**

**따라서 topModule.v를 run할 경우, logic\_gate.v을 그대로 인스턴스하게 되어 new\_result 값이 [1:0]new\_a + 3’b111로 실행된다.**