TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018-06-04 (66 회차)

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 정유경
ucong@naver.com

FPGA Programming, Simulation 방법 복습

1. Vivado WebPack 다운로드 및 설치

Vivado HLx 2017.4: All OS installer Single-File Download (TAR/GZIP - 16.17 GB)

https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2017-4.html

2. Vivado WebPACK, SDK, free IP에 대한 License 받기

Vivado License Manager에서 Obtain License > Connect Now > 로그인 > 기간 None인 라이선스 2개 선택 > Host Computer를 본인 컴퓨터 설정 > 계정에 등록된 이메일에서 첨부파일(Xilinx.lic) 다운로드 > 다시 License Manager에서 Copy License > Activate License > View License Status 탭에서 라이센스 등록상태 확인

3. Vivado를 열고 새 프로젝트를 생성한다

Create Project > Project Name: AND_GATE > RTL Project (Do not Specify Sources체크 해제) > Add Sources > File Name: AND_GATE.vhd File Type, Target Language: VHDL선택, Simulator Language: Mixed > Default Part: Zybo (Boards 옵션으로 추가하면 Constraints와 Pin Mappings를 포함함) > Define Modules: I/O Ports를 정의한다. (a,b: in / res: out) > Vivado의 전체 Design Flow를 확인할 수 있다.

4. AND_GATE.vhd 에서 VHDL Code를 작성한다.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity AND_GATE is
        Port(   a : in STD_LOGIC;
                b : in STD_LOGIC;
                res : out STD_LOGIC);
end AND_GATE;
```

```
architecture Behavioral of AND_GATE is

begin
        c <= a AND b;
end Behavioral;
```

*. IP Catalog : Libraries we can add to our design (라이선스를 가지고 있다면 License 탭에서 Included로 확인할 수 있고, License 없다면 Purchase로 표시되어 있다)
*. Simulation > RTL Analysis > Synthesis > Implementation (Constraints 추가해야 함) > Program and Debug 탭에서 Generate Bitstream
*. Generate Bitstream 까지 끝나면 Open Hardware Manager를 선택하여 우리가 만든 FPGA를 프로그램 한다

Run Simulation, Synthesis Code

5. Add Sorces > Simulation Sources 선택 > File Type: VHDL / Name: AND_tb / include all design sources 체크
6. Testbench Generator VHDL을 검색하여 아까 작성한 AND_GATE.vhd 파일을 복사 붙여넣기 > Generate VHDL Testbench > AND_tb.vhd 에 복사 붙여넣기 > Save
(https://www.doulos.com/knowhow/perl/testbench_creation/)

```
library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Std.all;
entity AND_GATE_tb is
end;
architecture bench of AND_GATE_tb is
  component AND_GATE
        Port(     a : in STD_LOGIC;
                  b : in STD_LOGIC;
        res : out STD_LOGIC);
```

```
  end component;
  signal a: STD_LOGIC;
  signal b: STD_LOGIC;
  signal res: STD_LOGIC;
begin
  uut: AND_GATE port map ( a     => a,
                           b     => b,
                           res => res );
  stimulus: process
  begin
    -- Put initialization code here
    -- Put test bench stimulus code here
    wait;
  end process;
end;
```

7. (ab : 00, 01, 10, 11의 모든 조합에 대하여 res를 확인하기 위해) initialization 코드를 작성한다. > Save

```
-- Put initialization code here
        a <= '0';
        b <= '0';
        wait for 10 ns;

        a <= '1';
        b <= '0';
        wait for 10 ns;
```

```
a <= '0';
b <= '1';
wait for 10 ns;

a <= '1';
b <= '1';
wait for 10 ns;

a <= '0';
b <= '0';
```

8. Synthesis해서 코드에 오류가 있는지 확인 > 성공했다는 메시지가 나오면 Open Implementation선택하지 않고 그냥 cancel
9. Run Behavioral Symulation > 상단의 Run for 탭에서 Simulation할 시간을 설정해준다(5 us)
10. Open Synthesized Design > Layout of the FPGA를 확인할 수 있다 > Synthesis의 Schematics 탭을 보면 우리가 작성한 코드로 생성된 회로를 볼 수 있다.

Implemetation Design and Create Bitstream

11. RTL Analysis > Open Elaborated Design 에서 a,b를 스위치에 res를 LED에 연결한다.(I/O Planning과 Constraint 설정을 해주는 것, 우리가 만든 a,b,res를 FPGA핀에 연결해준다) > Schematic 체크 > 상단 Layout > I/O Planning 체크
12. Add Sources > Create Constraints (constraints.xdc를 생성한다) > Elaborated Design이 out-of-date라면 reload한다
13. 하단의 I/O Ports에서 Direction, Site, Voltage level을 정해준다

LVCMOS33

https://reference.digilentinc.com/reference/programmable-logic/zybo/reference-manual 을 참조하여 어떤 핀을 사용할 것인지 결정한다 > Save > Selecte an existing xdc file 선택 > Constraints.xdc 파일 열어보면 자동 생성된 pin out을 확인할 수 있다.

Set_property IOSTANDARD LVCMOS33 {get_ports a}

set_property PACKAGE_PIN M16 {get_ports a} // M16을 c와 연결

14. Run Implementation (out-of-date이므로 synthesis해주고) > Open Implementation Design

Netlist에서 a, b ,res 선택하고 우클릭 Highlight 선택하면 Device탭에서 a,b,c가Hardware상에서 어떻게 연결되어 있는지 확인할 수 있다.
Implemented Design에서 여러가지를 확인할 수 있지만 우선 넘어가도록 한다.
14. Generate Bitstream
하단의 Design Runs에서 얼마나 진행되었는지 확인이 가능하다.

Open Hardware Manager and Programming FPGA
15. 생성한 Bitstream을 우리가 만든 FPGA에 다운로드한다.
Hardware Manager 클릭 > Open target > Auto Connect > Program Device에서 Bitstream File을 선택해주고 FPGA에 다운로드한다.
16. 스위치를 on/off 하여 led동작을 확인한다.

VHDL 기초
http://esd.cs.ucr.edu/labs/tutorial/
https://cdn.instructables.com/ORIG/FWB/2H79/IWMQTA48/FWB2H79IWMQTA48.pdf
http://users.wpi.edu/~rjduck/Vivado%20Simple%20VHDL%20Test%20Bench.pdf
http://snslab.kangwon.ac.kr/old/?doc=bbs/gbdownload.php&bo_table=pds&wr_id=113&index=1
Verilog 기초
https://www.xilinx.com/support/documentation/university/Vivado-Teaching/HDL-Design/2013x/Nexys4/Verilog/docs-pdf/Vivado_tutorial.pdf
http://users.wpi.edu/~rjduck/Vivado%20Simple%20Verilog%20Test%20Fixture.pdf
http://www.kocw.net/home/search/kemView.do?kemId=134122&ar=relateCourse

개념 이해하기
*. FPGA (field-programmable gate array)이미 설계된 하드웨어를 반도체로 생산하기 직전 최종적으로 하드웨어의 동작 및 성능을 검증하기 위해 제작하는 중간 개발물 형태의 집적 회로(IC). 회로 변경이 불가능한 일반 반도체와 달리 여러 번 회로를 다시 새겨 넣을 수 있는 반도체다. 오류 발생 시 수정이 가능하고 개발시간이 짧으며 초기 개발비용이 적게 든다. 그러나 일반적으로 속도가 느리고 복잡한 설계에 적용이 불가하며 소비전력이 크다는 단점이 있다. 반도체 제조업자 측에서 보면 양산되어 일반적 용도로 사용되므로 범용 IC의 범주에 속하고, 사용자 측에서 보면 사용자 요구

에 맞게 프로그래밍하여 사용할 수 있으므로 주문형 반도체(ASIC) 범주에 속한다.

AND, OR, NOT, NAND, NOR, XOR, XNOR, Buffer, 가산기와 같은 논리 IC라고 한다. (Logic IC) 내부구성에 따라 RTL, DTL, TTL, CMOS, ECL등으로 나뉘며 모든 소자는 반도체가 들어간다. ECL로 설계했을 때 가장 속도가 빠르다. ECL은 반도체를 포화시키지 않기 때문에 일반적인 회로보다 훨씬 속도가 높으며 보통 수백 Mhz ~ 수 Ghz이상은 모두 ECL로 구성된다. 그 다음은 TTL > CMOS 순이다.

TTL과 CMOS를 구분해보면 다음 표와 같다.

	TTL	**CMOS**
사용 전압	5V	2V ~ 6V
소비 전력	많다	아주작다
공급 가능 전류	20mA	10mA

TTL은 BJT와 diode로 구성된 Lgic IC를 말한다. 이 TTL의 주파수 및 전력특성을 개선한 것이 CMOS (Complementary Metal Oxide Semiconductor)이며, CMOS는 NMOS와 PMOSFET로 구성된 Logic IC이다. 여러 계열의 패밀리를 가지고 있으며 HC,HCT 등의 타입이 있다.

논리 IC를 구분하는 방법은 다음 표와 같다.

TTL	74LS__, 74AS__, 74F__, 54LS__ (또는 영문접두어 없이 7406)
CMOS	74HC__, 74HCT__, 74AC, 74ACT, 74FCT

- Resistor Transistor Logic(RTL): 저항-트랜지스터 논리, 저항과 트랜지스터로 구성하는 회로이다.
- Diode Transistor Logic(DTL): 다이오드-트랜지스터 회로이다. 이 회로는 다이오드의 AND 회로, OR 회로에 트랜지스터의 NOT 회로를 종합한 것이며, 안정성이 높다는 특징이 있다.
- Transistor-Transistor Logic(TTL): 트랜지스터-트랜지스터 논리, 트랜지스터 회로이다. 이 회로는 DTL의 다이오드 대신에 멀티 이미터 트랜지스터를 사용한 것이며, IC로서의 구성이 간단하고, 저전력 고속동작 등의 장점이 있다.