Hardware System Design 4190.309A
Seoul National University

Practice #7. How to use FPGA board
Jiwon Lee, Sangjun Son

Goal

- ZED Board Tutorial.
  - Setup the board.
  - *SW2LED* module: A combinational logic that blinks `[7:0]LED` in response to `[7:0]SWITCH`.
- Implement a simple sequential logic with external clock.

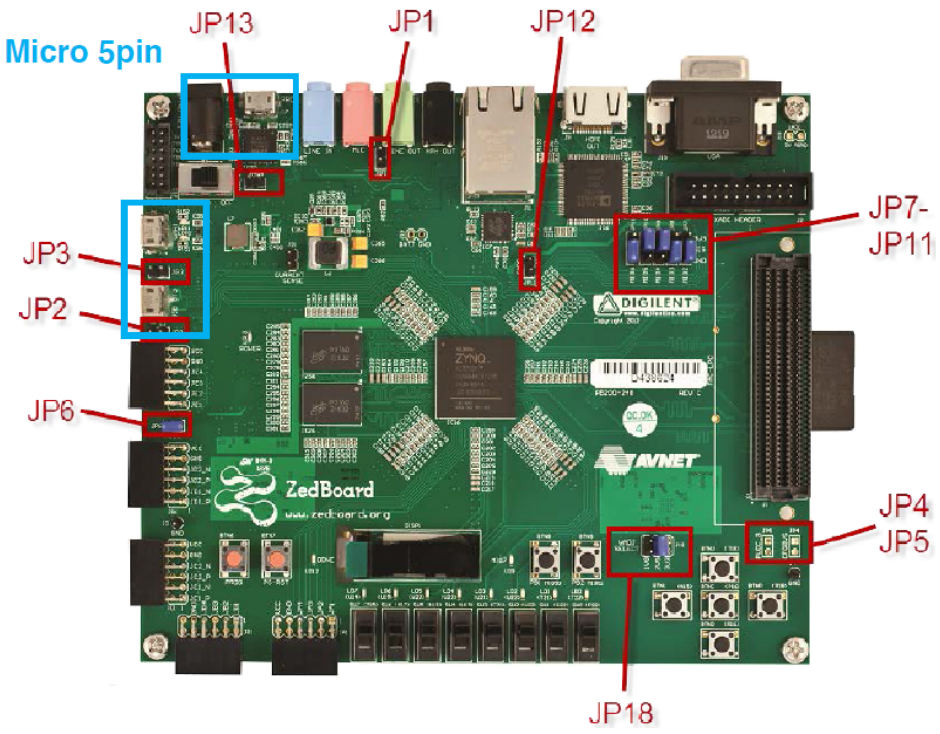

Figure 1: ZedBoard Jumper Map [1].

1 Implementation

이번 프로젝트는 ZED Board를 처음으로 수령하고 PC에 연결하여 각각의 컴포넌트를 조작할 수 있는 환경설정을 목표로 한다. 이전 Lab 세션에서는 Design과 Simulation을 통해 Verilog 코드를 작성하고 테스트 벤치 상에서 알맞는 동작을 하는지 간접적으로 확인하는 작업을 해왔다. 이번에는 Synthesis와 Implementation을 한 이후에 Board에 Bitstream Generator를 이용해 실질적으로 올바른 동작을 하는지 눈으로 확인할 수 있을 것이다. 튜터리얼로 토글된 스위치에 해당하는 위치의 LED가 켜지는 프로그램을 통해 보드의 컴포넌트 작동을 확인하고 이후 실습으로 *1 sec Checker*를 구현하고 이를 보드 위에서 잘 작동하는지 확인할 것이다.

1.1 ZED Board Tutorial

8 switches와 8 LEDs를 서로 연결하는 코드는 다음과 같으며 보드에 해당하는 컴포넌트 번호는 Hardware User's Guide [2]를 참고하였다. 기존 프로젝트와는 다르게 User Constraints를 작성해주어야 하며 이는 제공된 템플릿을

활용해 원하는 포트 및 전달받을 신호 변수를 지정해주었다.

SW2LED

- Verilog Code, `sw2led.v`

```
1  `timescale 1ns / 1ps
2  module sw2led(
3      input [7:0] SW,
4      output [7:0] LD
5      );
6      assign LD = SW;
7
8  endmodule
```

- User Constraints, `sw2led.xdc`

```
1  # ------------------------------------------------------------------------
2  # User DIP Switches - Bank 35
3  # ------------------------------------------------------------------------
4  set_property PACKAGE_PIN F22 [get_ports {SW[0]}];  # "SW0"
5  set_property PACKAGE_PIN G22 [get_ports {SW[1]}];  # "SW1"
6  set_property PACKAGE_PIN H22 [get_ports {SW[2]}];  # "SW2"
7  set_property PACKAGE_PIN F21 [get_ports {SW[3]}];  # "SW3"
8  set_property PACKAGE_PIN H19 [get_ports {SW[4]}];  # "SW4"
9  set_property PACKAGE_PIN H18 [get_ports {SW[5]}];  # "SW5"
10 set_property PACKAGE_PIN H17 [get_ports {SW[6]}];  # "SW6"
11 set_property PACKAGE_PIN M15 [get_ports {SW[7]}];  # "SW7"
12 set_property IOSTANDARD LVCMOS25 [get_ports -of_objects [get_iobanks 35]];
13
14 # ------------------------------------------------------------------------
15 # User LEDs - Bank 33
16 # ------------------------------------------------------------------------
17 set_property PACKAGE_PIN T22 [get_ports {LD[0]}];  # "LD0"
18 set_property PACKAGE_PIN T21 [get_ports {LD[1]}];  # "LD1"
19 set_property PACKAGE_PIN U22 [get_ports {LD[2]}];  # "LD2"
20 set_property PACKAGE_PIN U21 [get_ports {LD[3]}];  # "LD3"
21 set_property PACKAGE_PIN V22 [get_ports {LD[4]}];  # "LD4"
22 set_property PACKAGE_PIN W22 [get_ports {LD[5]}];  # "LD5"
23 set_property PACKAGE_PIN U19 [get_ports {LD[6]}];  # "LD6"
24 set_property PACKAGE_PIN U14 [get_ports {LD[7]}];  # "LD7"
25 set_property IOSTANDARD LVCMOS33 [get_ports -of_objects [get_iobanks 33]];
26 set_property IOSTANDARD LVCMOS25 [get_ports -of_objects [get_iobanks 34]];
```

위의 모듈 코드를 보면 입력과 출력으로 8 비트 `SW`와 `LD`가 있는 것을 확인할 수 있다. 단순히 두 개의 wire를 이어주는 논리로 작성이 되었으며 사용되는 입력과 출력 신호는 각각 DIP Switches (Bank 35)와 User LEDs (Bank 33)에 해당하는 컴포넌트의 포트번호를 통해 연결되어 있는 것을 확인할 수 있다 [2].

1.2 Practice: One Second Checker

One Second Checker는 크게 Down counter와 Up counter로 구성되어 있다. Down counter는 `GCLK`의 신호를 초 단위로 바꿔주는 역할을 하며 Up counter는 Down counter에 따라 즉 초 단위로 카운트가 되는 역할을 한다. Up counter를 LED에 연결하여 보드에서 확인할 수 있게 하였다.

One Second Checker

- Verilog Code, `one_sec_checker.v`

```
`timescale 1ns / 1ps

module one_sec_checker#(
    parameter CLK_FREQ = 28'd100000000
)(
    input GCLK,
    input BTNC,
    output [7:0] LD
    );

    reg [27:0] down_counter;
    reg [7:0] up_counter;
    assign LD = up_counter;

    initial begin
        up_counter <= 0;
        down_counter <= CLK_FREQ;
    end

    always @(posedge GCLK or posedge BTNC) begin
        if (BTNC) begin
            up_counter <= 0;
            down_counter <= CLK_FREQ;
        end
        else begin
            if (down_counter == 0) begin
                up_counter <= up_counter + 8'd1;
                down_counter <= CLK_FREQ;
            end
            else begin
                down_counter <= down_counter - 28'd1;
            end
        end
    end
endmodule
```

- User Constraints, `one_sec_checker.xdc`

```
# ----------------------------------------------------------------------------
# Clock Source - Bank 13
# ----------------------------------------------------------------------------
set_property PACKAGE_PIN Y9 [get_ports {GCLK}];  # "GCLK"
set_property IOSTANDARD LVCMOS33 [get_ports -of_objects [get_iobanks 13]];

# ----------------------------------------------------------------------------
# User LEDs - Bank 33
# ----------------------------------------------------------------------------
set_property PACKAGE_PIN T22 [get_ports {LD[0]}];  # "LD0"
set_property PACKAGE_PIN T21 [get_ports {LD[1]}];  # "LD1"
set_property PACKAGE_PIN U22 [get_ports {LD[2]}];  # "LD2"
set_property PACKAGE_PIN U21 [get_ports {LD[3]}];  # "LD3"
set_property PACKAGE_PIN V22 [get_ports {LD[4]}];  # "LD4"
set_property PACKAGE_PIN W22 [get_ports {LD[5]}];  # "LD5"
set_property PACKAGE_PIN U19 [get_ports {LD[6]}];  # "LD6"
set_property PACKAGE_PIN U14 [get_ports {LD[7]}];  # "LD7"
set_property IOSTANDARD LVCMOS33 [get_ports -of_objects [get_iobanks 33]];

# ----------------------------------------------------------------------------
# User Push Buttons - Bank 34
```

```
22  # ----------------------------------------------------------------------------
23  set_property PACKAGE_PIN P16 [get_ports {BTNC}];  # "BTNC"
24  set_property IOSTANDARD LVCMOS25 [get_ports -of_objects [get_iobanks 34]];
```

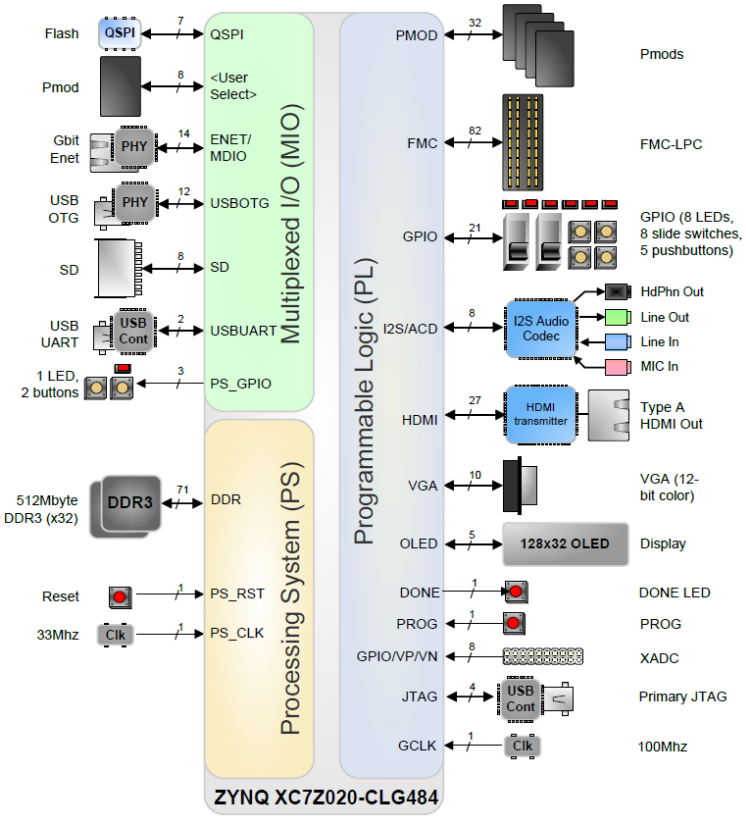

Figure 2: ZedBoard Block Diagram [2].

ZedBoard의 Clock frequency를 확인하기 위해서 User Guide에 적힌 스펙을 확인하면 된다. Figure 2에서 볼 수 있듯이 `GCLK`은 100 MHz의 주기를 가지므로 Down counter를 `CLK_FREQ` $= 10^8$으로 초기화 한다. 그리고 매 `GCLK` 신호 마다 counter에서 1을 빼주는 방식으로 업데이트 해준다. Down counter가 0이 됐다면 이는 1초가 흘렀음을 의미하고 Up counter (1초마다 증가하는 카운터)에 1을 증가시키고 동시에 Down counter를 다시 `CLK_FREQ` $= 10^8$으로 초기화 한다. 이때 리셋 신호인 `BTNC` 신호가 들어오면 Clock 신호에 Asynchronous하게 Down counter, Up counter 모두 초기화 해준다.

위의 User Constraints는 입력 `GCLK`, `BTNC`와 출력 `LD`에 해당하는 포트에 대응되는 환경 변수를 표현한 코드이. 각각 Clock Source (Bank 13), User Push Buttons (Bank 34) 와 User LEDs (Bank 33)에 해당하는 컴포넌트의 포트번호를 통해 연결되어 있는 것을 확인할 수 있다 [2].

2 Experiments

2.1 Implementation Results

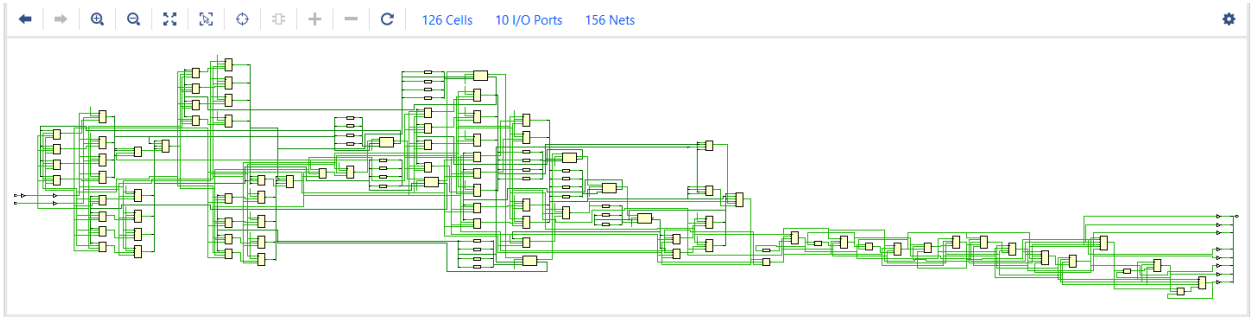

Figure 3: Schematic of *One Second Checker* after Implementation

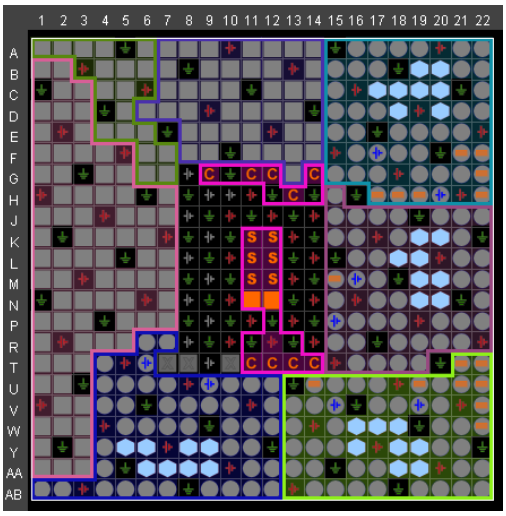

Figure 4: I/O ports used in *One Second Checker* after Implementation

2.2 Execution Results

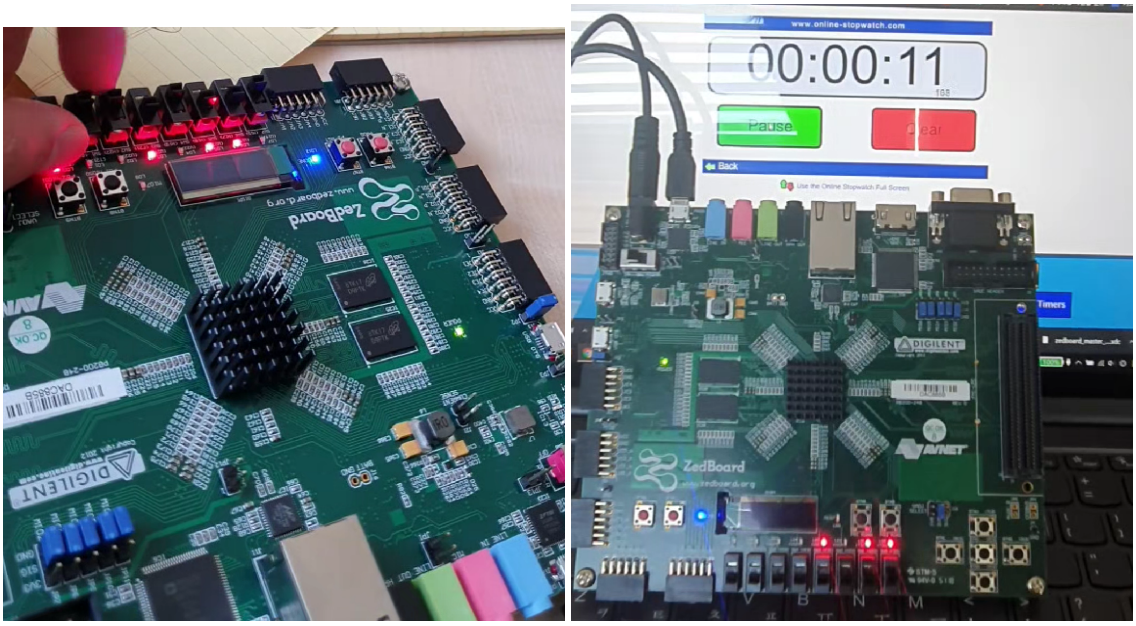

Figure 5: Execution results of *SW2LED* and *One Second Checker*

3 Conclusion

이번 주 실습 결과는 영상으로 촬영하여 압축파일에 함께 첨부하였다. 이번 실습은 구현 사항에 있어서는 그렇게 어려운 부분은 없었지만 기존의 실습과는 다르게 환경을 제대로 설정해주어야 원하는 결과를 얻을 수 있다는 것을 확인하였다.

Verilog 코드의 Port와 보드의 구성품과 연결하고 올바르게 전압을 올려줘야 하는 과정에서 혼란을 겪었고 앞으로의 실습에서도 사소한 구현이 동작여부를 좌지우지할 수 있을 것이라는 우려를 낳았다. ZedBoard User Guide [2]를 자주 참고할 것 같다. Synthesis, Implementation 그리고 Bitstream Generation을 수행하는데 오랜 시간이 걸리는 만큼 Simulation test bench의 중요성 또한 상기하였다.

References

[1] Computing Memory Architecture Lab. *Practice 5: PE implementation and BRAM modeling*. Hardware System Design, April 2021.

[2] ZedBoard. *Hardware User's Guide, Version 2.2*. Zynq™ Evaluation and Development, January 2014.