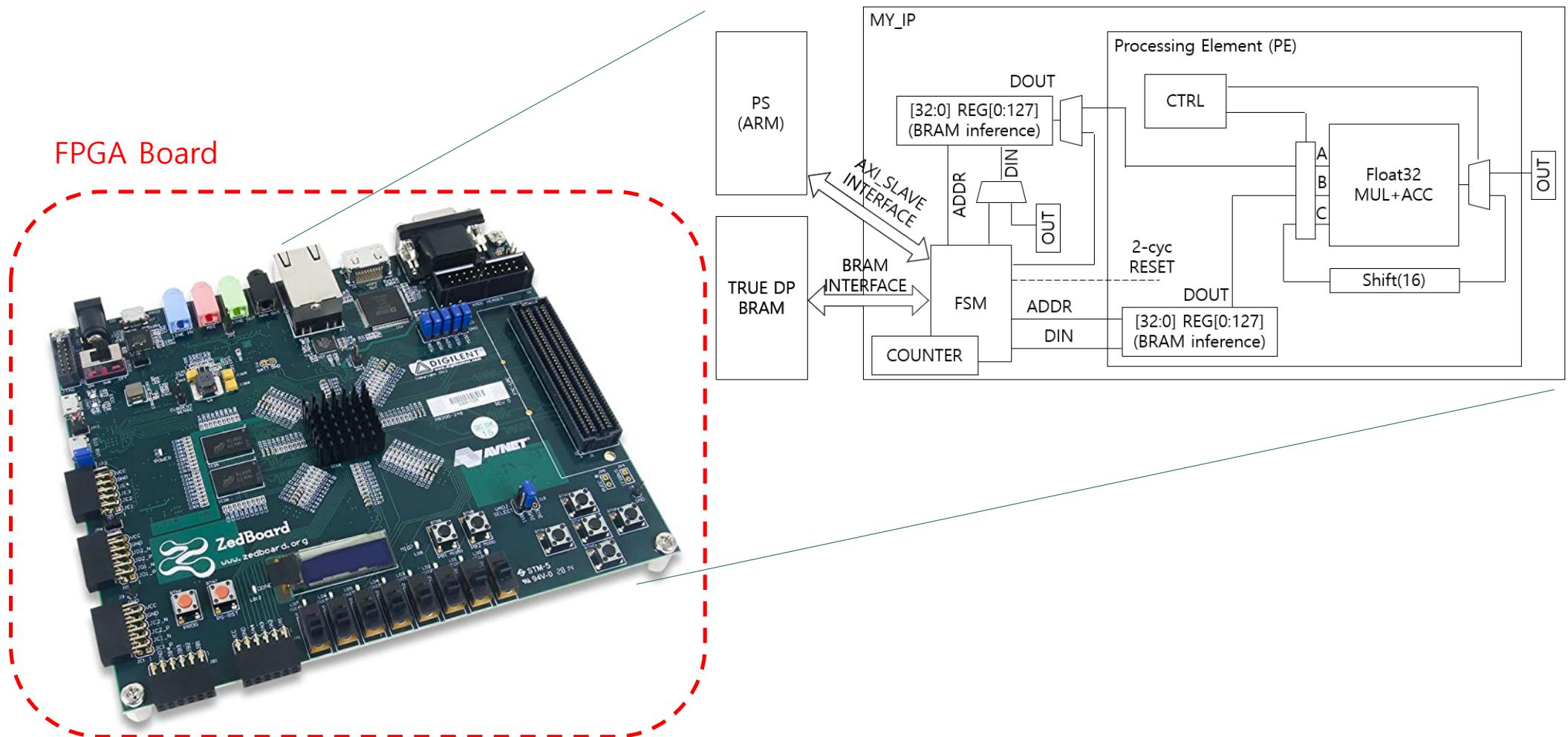


# Practice 8

- How to use FPGA board

Computing Memory Architecture Lab.

# Final Project Overview: Matrix Multiplication IP



# Overview

---

- **ZED Board Tutorial**

- Setup the board
- SW2LED module
  - A combinational logic that blinks [7:0] LED in response to [7:0] SWITCH

- **Practice**

- Implement a simple sequential logic with external clock

# ZED Board Tutorial

# Notations

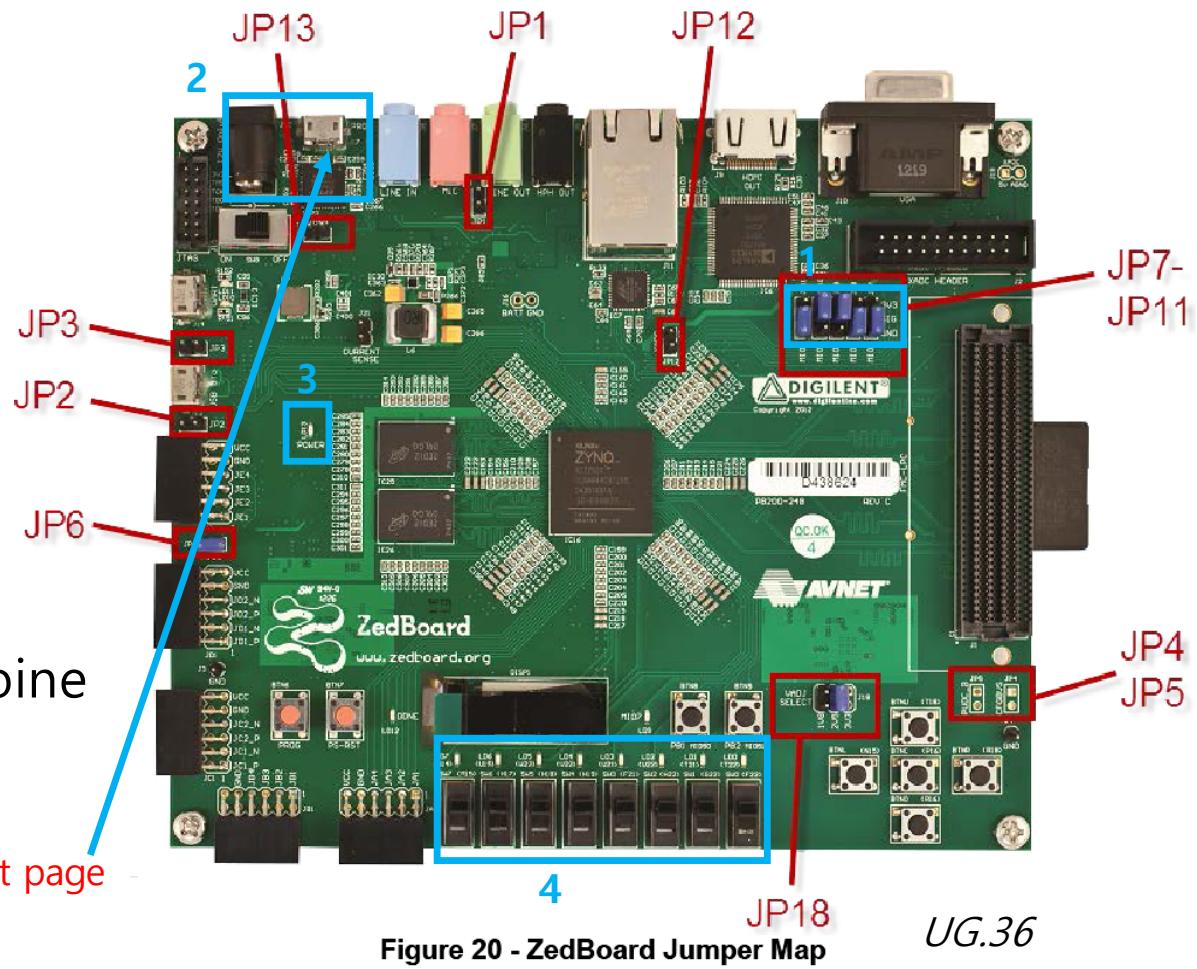
---

- UG.XXX (User Guide)
  - Page XXX of ZedBoard\_HW\_UG\_v2\_2.pdf
    - With respect to *the PDF page number* not footer
- SM.YYY (Schematic)
  - Page YYY of ZedBoard\_RevD.2\_Schematic\_130516.pdf
- DC.LLL (Design Constraint)
  - Line LLL of zedboard\_master\_XDC\_RevC\_D\_v3.xdc
- BD.PPP (Board)
  - Part PPP of the board

# Preparing the Board

- Set JTAG boot mode (1)
  - BD.JP7 ~ BD.JP11
    - 5'b01100 -> 5'b00000
- Insert USB-JTAG cable (2)
  - BD.J17
- Insert power cable (2)
  - BD.J20, BD.SW8
  - Power ON -> see BD.LD13 (3)
- SW/LED (4)
  - Check locations of today's heroine
  - BD.SW0~7, BD.LD0~7

Cautions on next page



# Cautions for the Board

- Micro 5pin slots
  - Well known for breaking down.
    - Weak soldering
    - Stiff
    - Structural deficiency
  - If the cable connection is stiff, press the hook with a knife tip
  
- No extra board available
  - No further labs (unable to do without board)

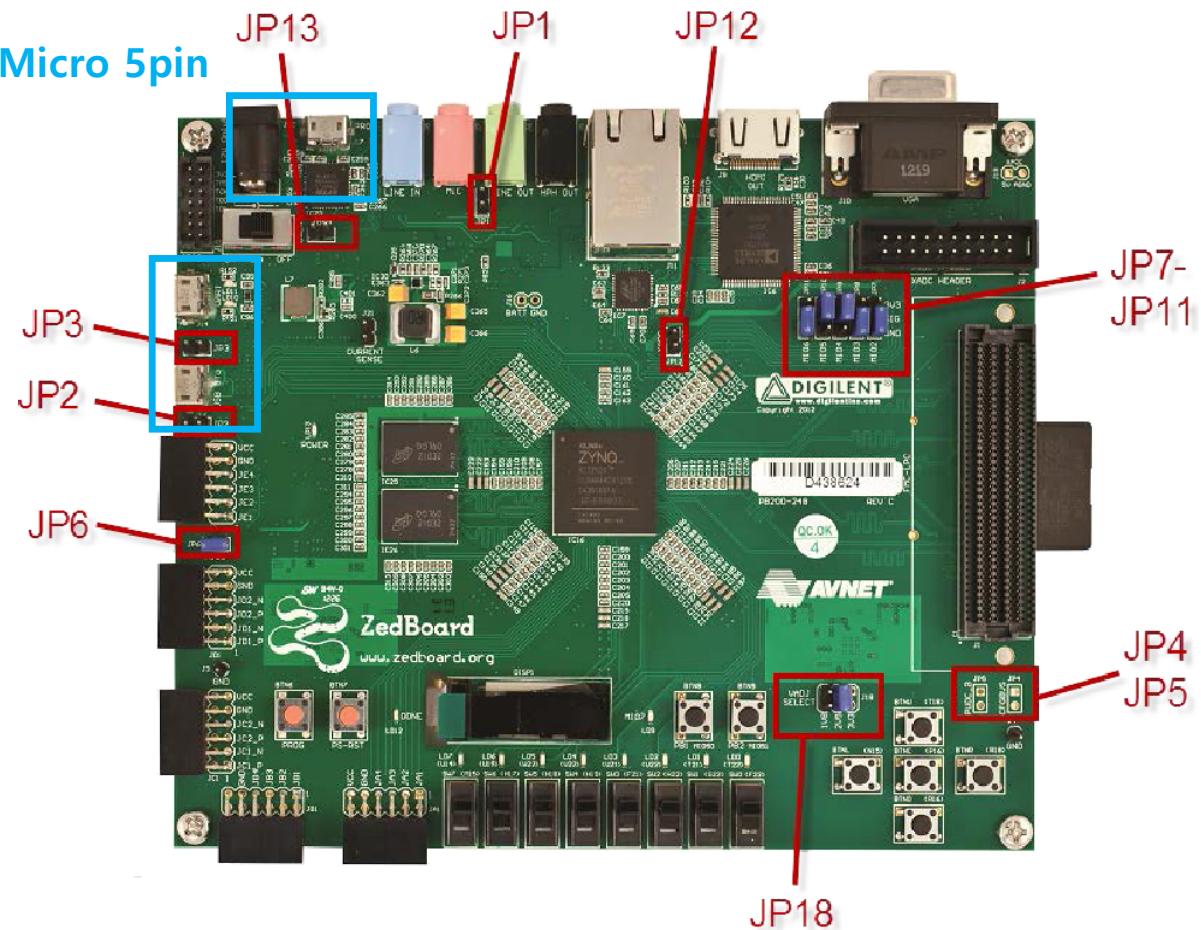
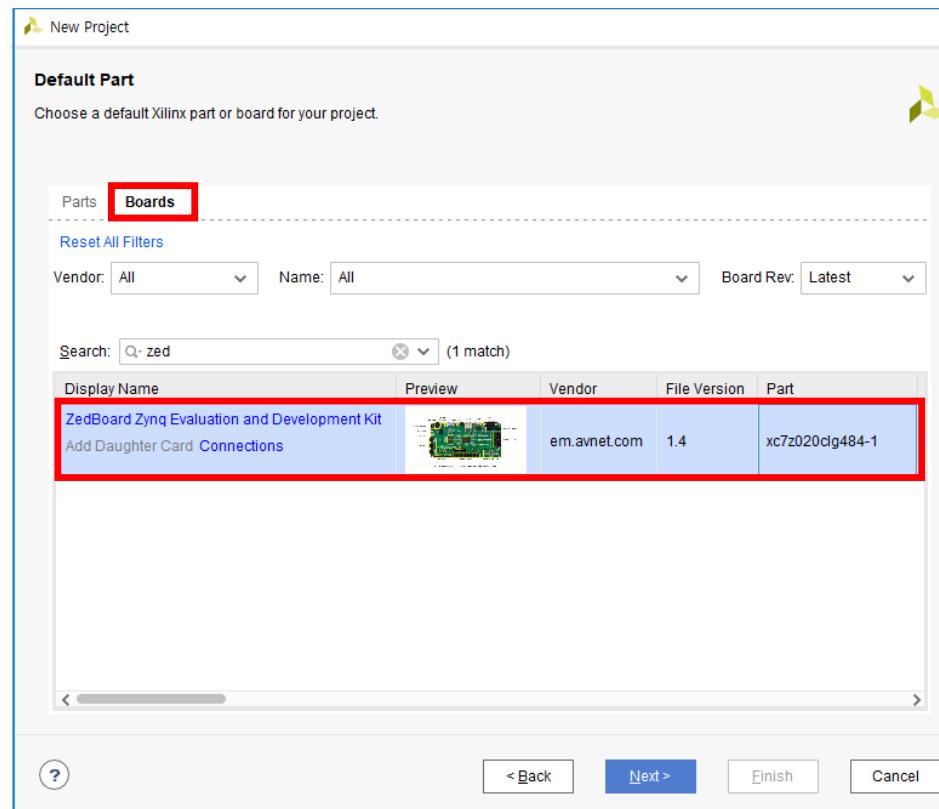


Figure 20 - ZedBoard Jumper Map

# Vivado Project Creation

- Create Project - Choose part or board
  - ZedBoard



# SW2LED Module

## ■ Edit your design

### - A Verilog file (1)

- Add or create **design sources**
- Simple SW+LED module
  - [7:0] LD = [7:0] SW;

### - An XDC (constraint) file (2)

#### • Add or create **constraints**

- 8-switches
  - PACKAGE\_PIN: UG.20, DC.237, SM.9
  - IOSTANDARD: UG.5, DC.371, BDJ18
- 8-LEDs
  - PACKAGE\_PIN: UG.21, DC.175, SM.9
  - IOSTANDARD: UG.5, DC.362/367

The screenshot shows the Xilinx Project Manager interface with the following components:

- Sources Panel:** Shows the project structure with "Design Sources (1)" containing "sw2led (sw2led.v)".
- Verilog Editor:** Displays the Verilog source code for the "sw2led" module:

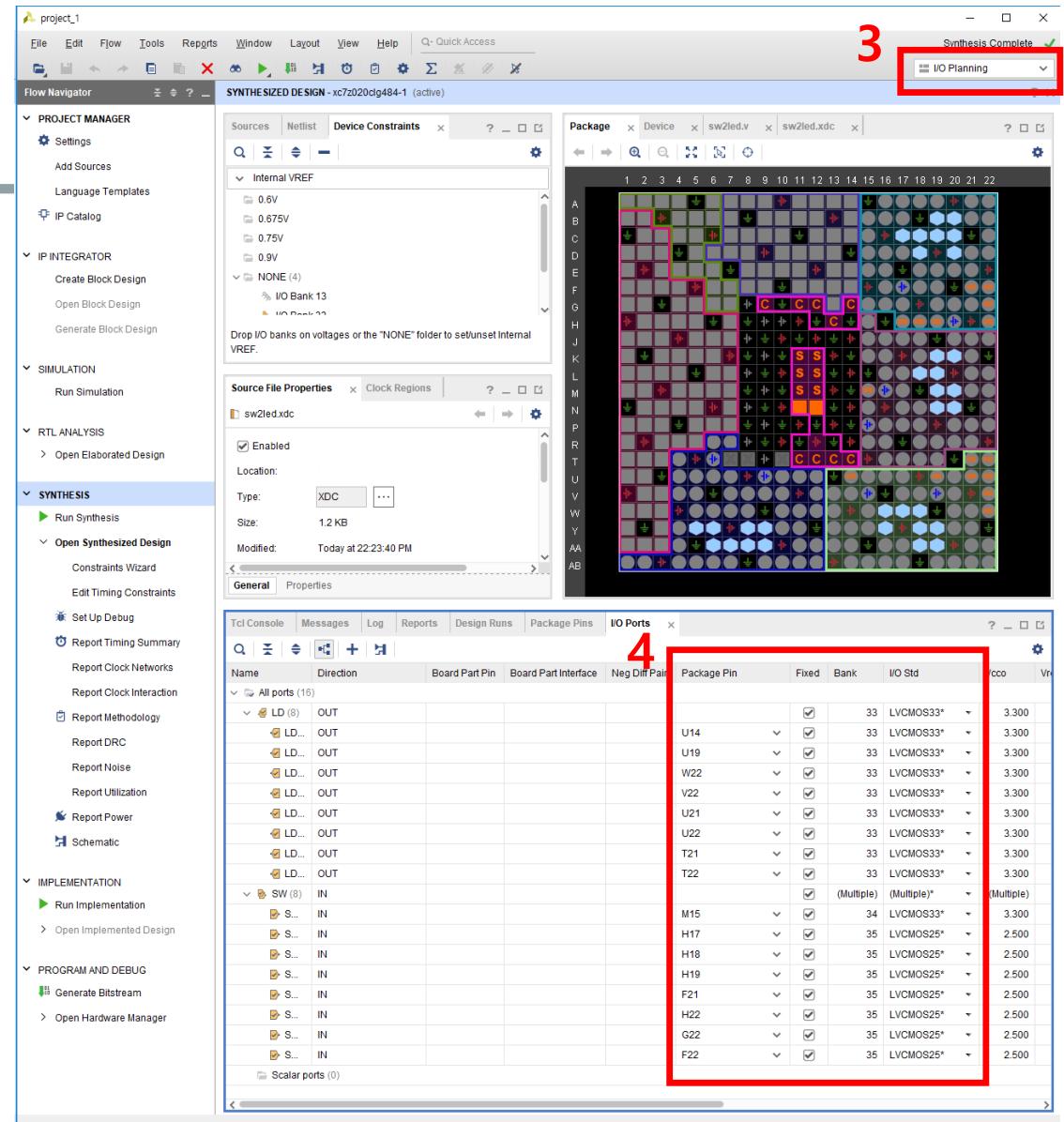
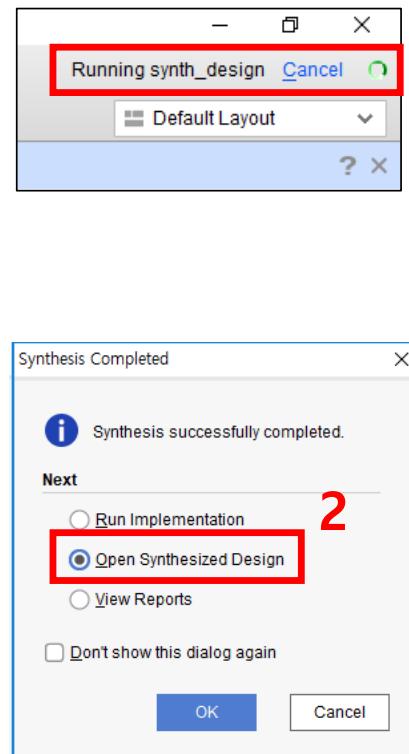
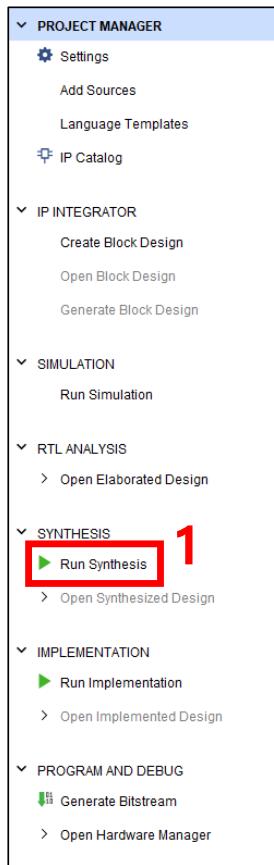
```
1 `timescale 1ns / 1ps
2
3 module sw2led(
4     input [7:0] SW,
5     output [7:0] LD
6 );
7
8 assign LD = SW;
9 endmodule
10
```
- XDC Editor:** Displays the XDC constraint file "sw2led.xdc":

```
1 set_property PACKAGE_PIN F22 [get_ports {SW[0]}]; # "S0"
2 set_property PACKAGE_PIN B22 [get_ports {SW[1]}]; # "S1"
3 set_property PACKAGE_PIN H22 [get_ports {SW[2]}]; # "S2"
4 set_property PACKAGE_PIN F21 [get_ports {SW[3]}]; # "S3"
5 set_property PACKAGE_PIN H19 [get_ports {SW[4]}]; # "S4"
6 set_property PACKAGE_PIN H18 [get_ports {SW[5]}]; # "S5"
7 set_property PACKAGE_PIN H17 [get_ports {SW[6]}]; # "S6"
8 set_property PACKAGE_PIN M15 [get_ports {SW[7]}]; # "S7"
9 set_property IOSTANDARD LVCMSOS25 [get_ports -of_objects {get_lobanks 35}];
10
11 set_property PACKAGE_PIN T22 [get_ports {LD[0]}]; # "L0"
12 set_property PACKAGE_PIN T21 [get_ports {LD[1]}]; # "L1"
13 set_property PACKAGE_PIN U22 [get_ports {LD[2]}]; # "L2"
14 set_property PACKAGE_PIN U21 [get_ports {LD[3]}]; # "L3"
15 set_property PACKAGE_PIN V22 [get_ports {LD[4]}]; # "L4"
16 set_property PACKAGE_PIN V21 [get_ports {LD[5]}]; # "L5"
17 set_property PACKAGE_PIN U19 [get_ports {LD[6]}]; # "L6"
18 set_property PACKAGE_PIN U14 [get_ports {LD[7]}]; # "L7"
19 set_property IOSTANDARD LVCMSOS33 [get_ports -of_objects {get_lobanks 33}];
20 set_property IOSTANDARD LVCMSOS33 [get_ports -of_objects {get_lobanks 34}];
```
- Properties Panel:** Shows the properties for the "sw2led.xdc" file, including "Enabled" checked, "Location" as C:/Users/rflagk/OneDrive/project\_1/project\_1.srcs/constrs\_1/new/sw2led.xdc, and various port settings.

34번 뱅크 2.5V  
#34 bank 2.5V

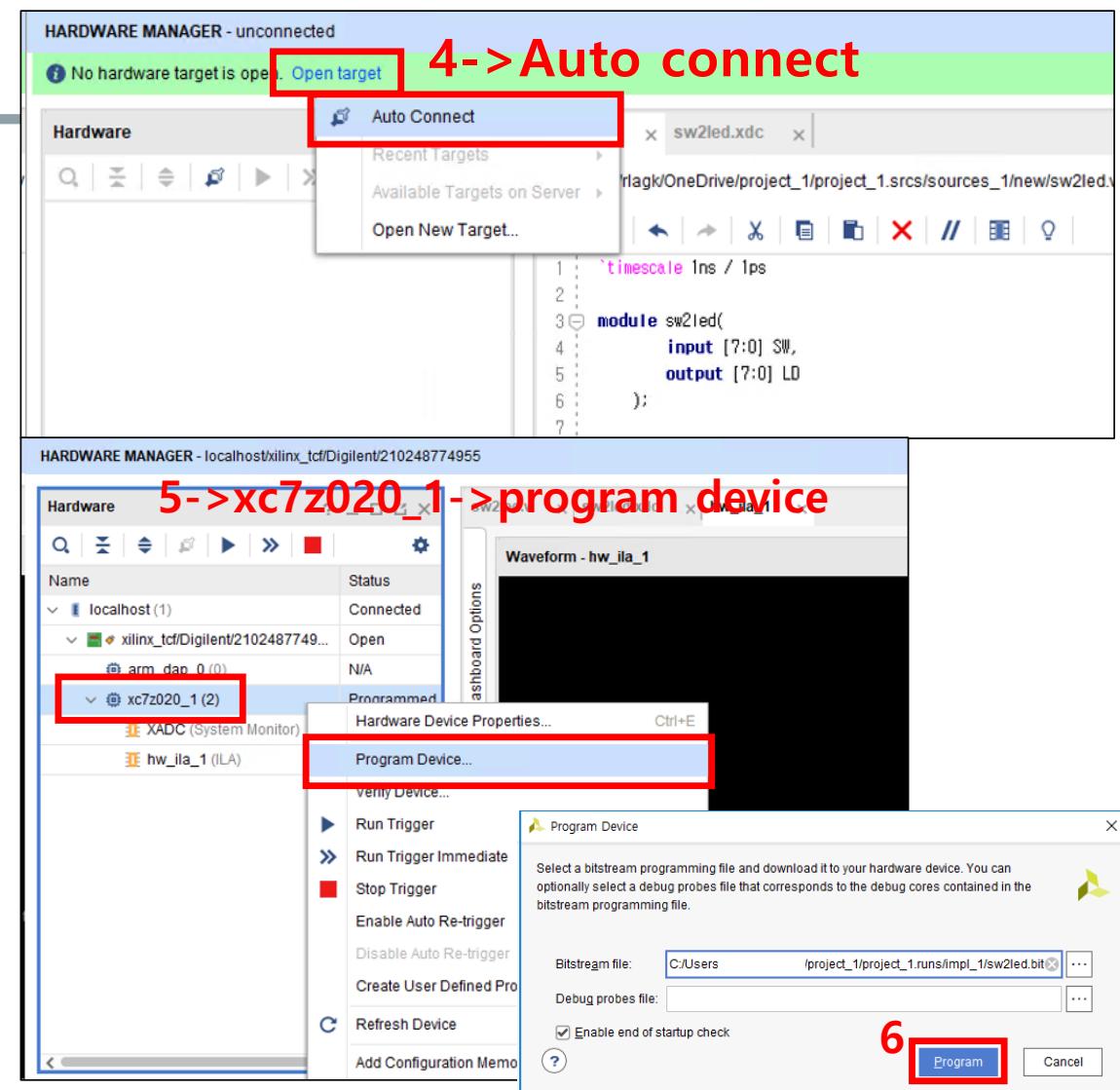
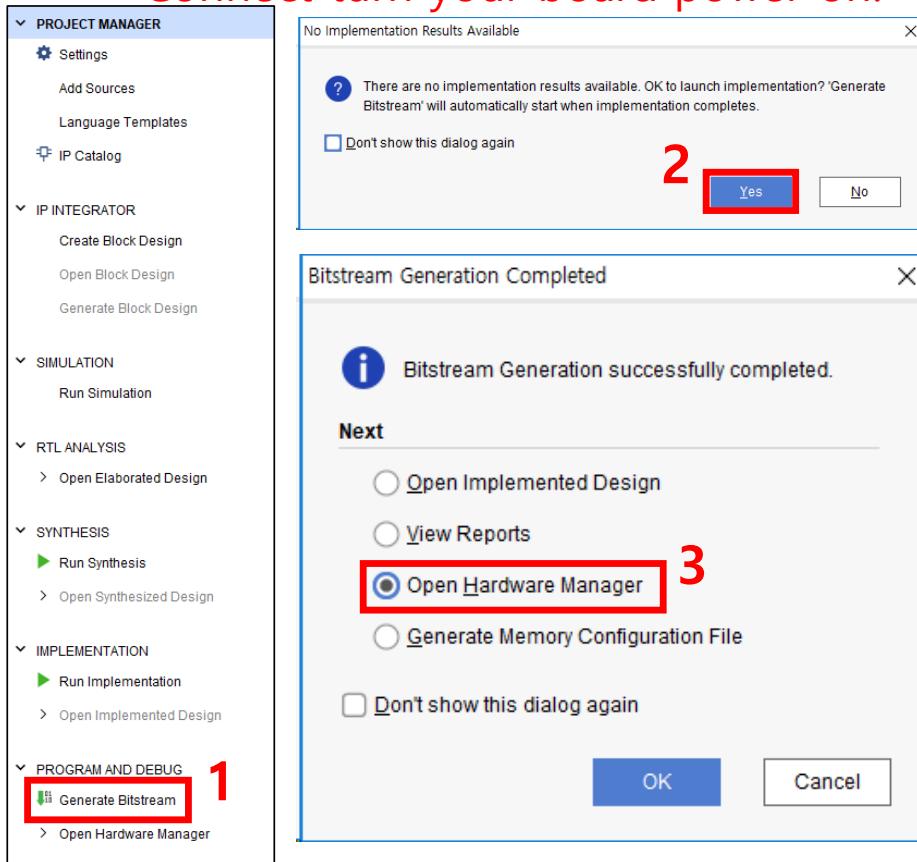
# Synthesis

## ▪ Synthesize and check



# Implementation

- Implement your design into FPGA
  - Connect turn your board power on.



# Enjoy Your Design!

- Toggle BD.SW0~7
  - And see BD.LD0~7

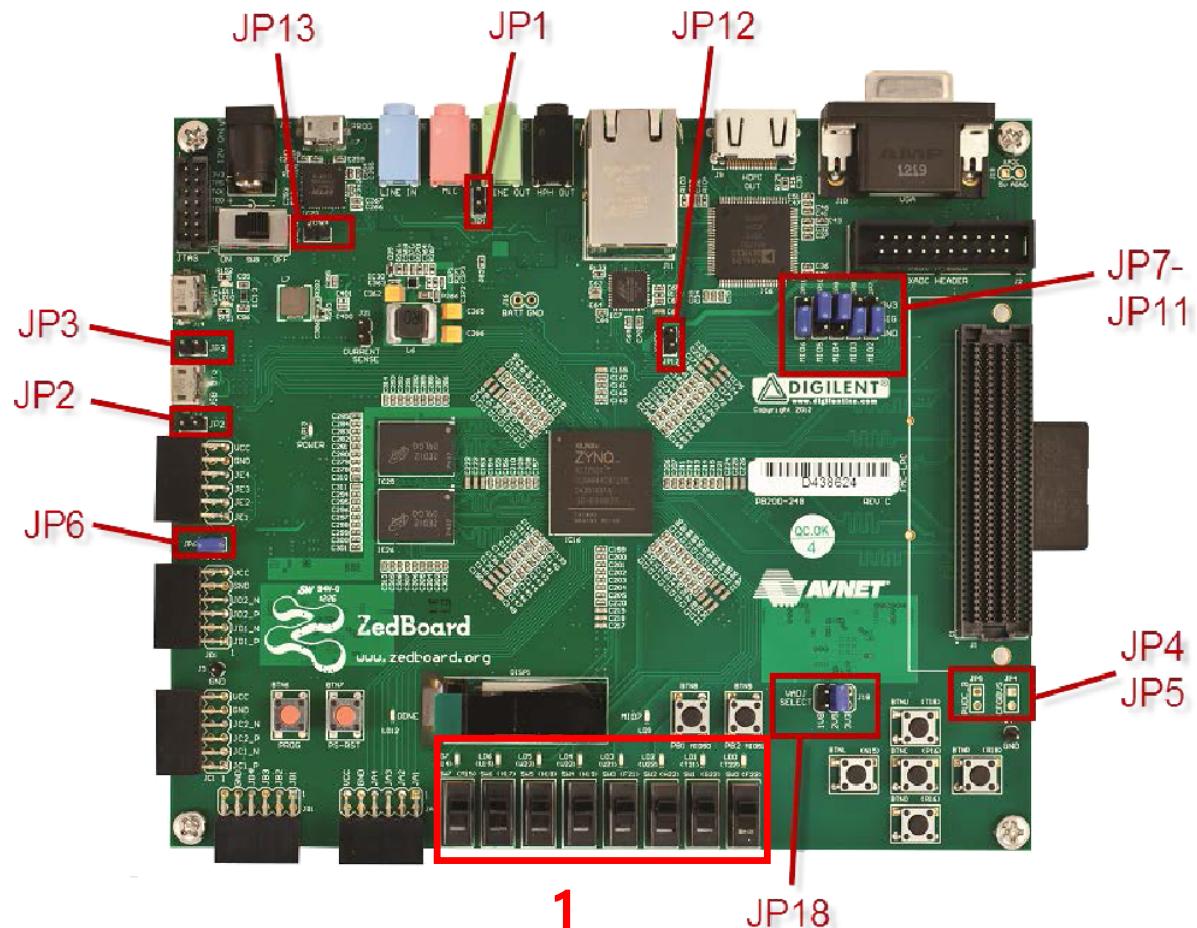
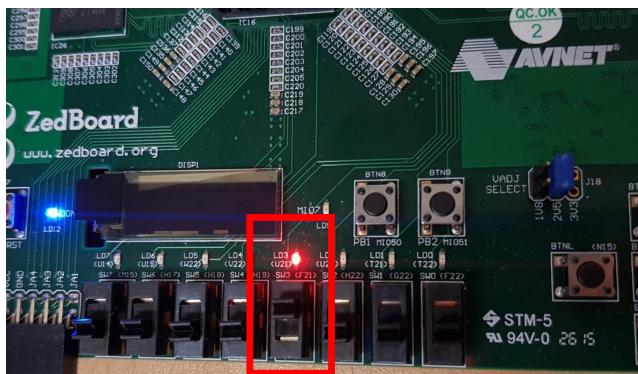


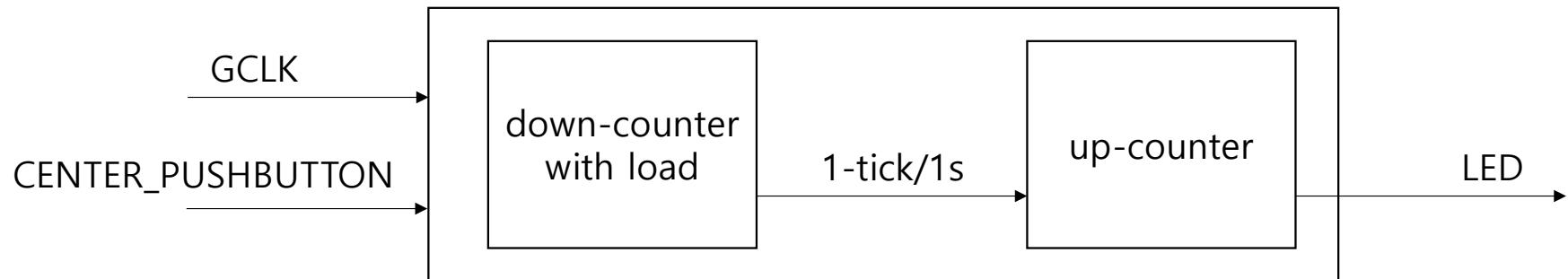
Figure 20 - ZedBoard Jumper Map

# Main Practice

# Practice

## ■ Implementing a simple sequential logic (1-sec checker)

- Implement a "1-sec checker" working with your FPGA board.



- 1-sec checker requirements
  - Use GCLK as an input clock (UG.4) and manipulate it to count 1s
    - Build an additional down-counter
  - Design a simple up-counter ticking every 1-second
    - Use [7:0] LED to visualize the counter value
  - Use CENTER\_PUSHBUTTON to assert synchronous reset (UG.4)

# Don't Break ZedBoard

- We have no extra ZedBoard to give students who break their ZedBoard
- So please use your ZedBoard carefully

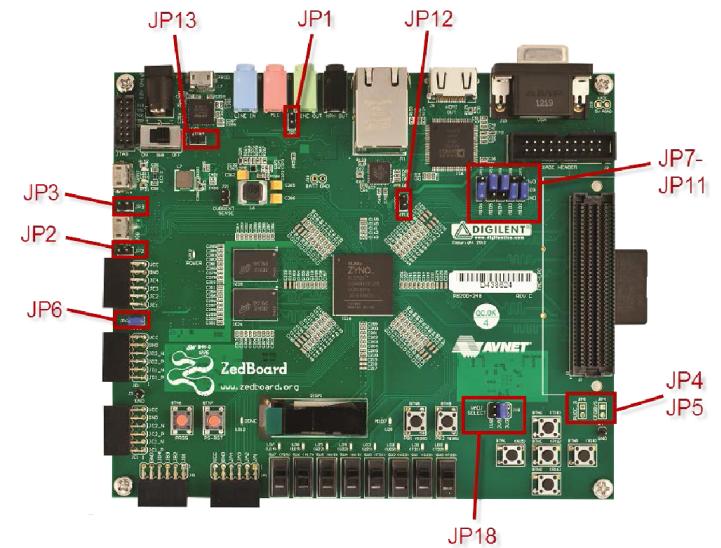


Figure 20 - ZedBoard Jumper Map

# Homework

---

## ■ Requirements

- Result
  - Attach your project folder with all your verilog codes (1-sec checker)
  - Attach a video that can show your "1-sec checker" works with LED
- Report
  - Explain "1-sec checker" that you implemented
  - In your own words
  - Either in Korean or in English
  - # of pages does not matter
  - **PDF only!!**
- **Result + Report to a .zip**

## ■ Upload (.zip) file on ETL

- Submit one (.zip) file
  - zip file name : [Lab08]name.zip (ex : [Lab08]홍길동.zip)
- Due: 5/12(TUE) 23:59
  - **No Late Submission**