香港中文大學
The Chinese University of Hong Kong

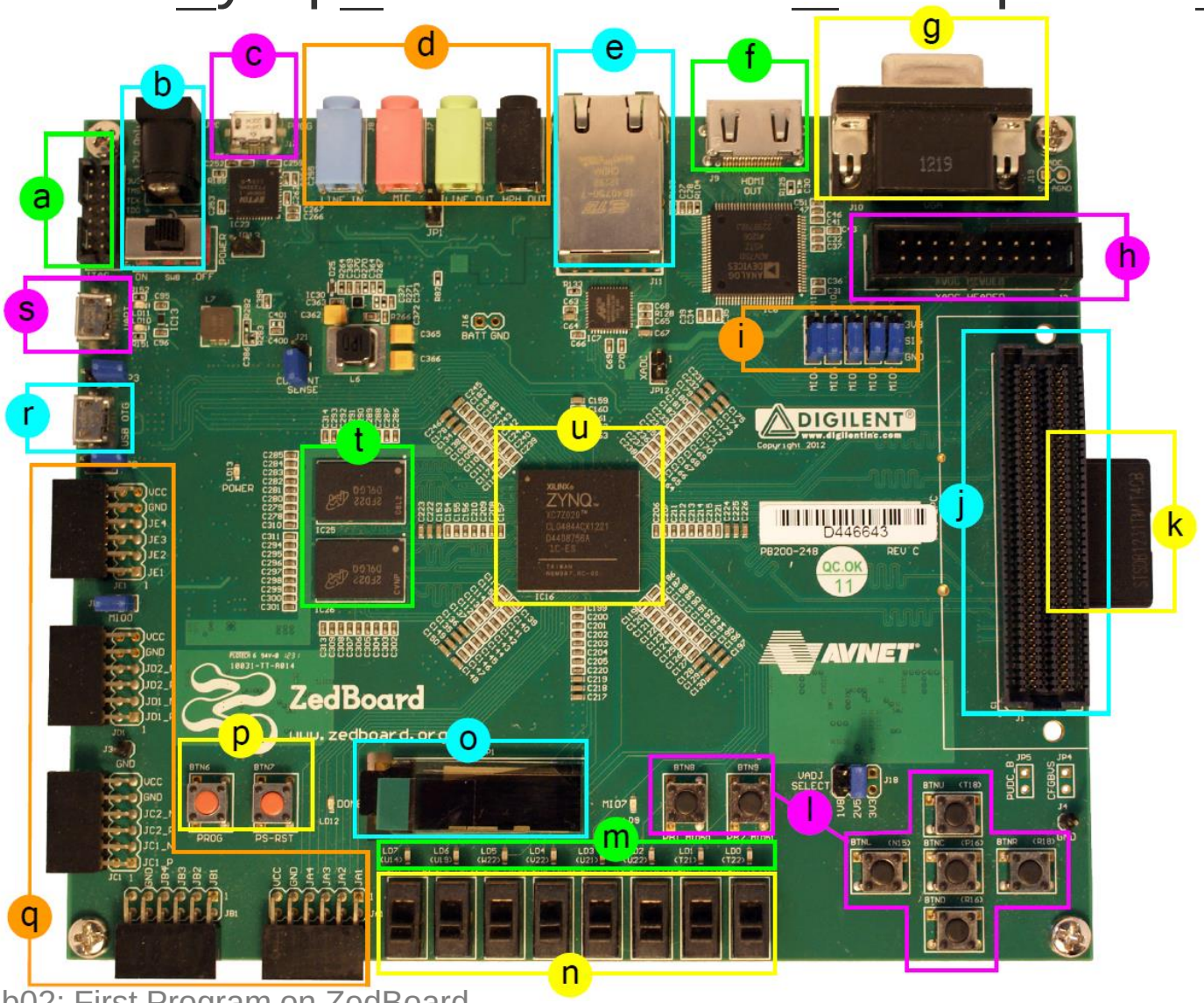
CENG3430 Rapid Prototyping of Digital Systems

Lab02:
First Program on ZedBoard

Setup for Zedboard

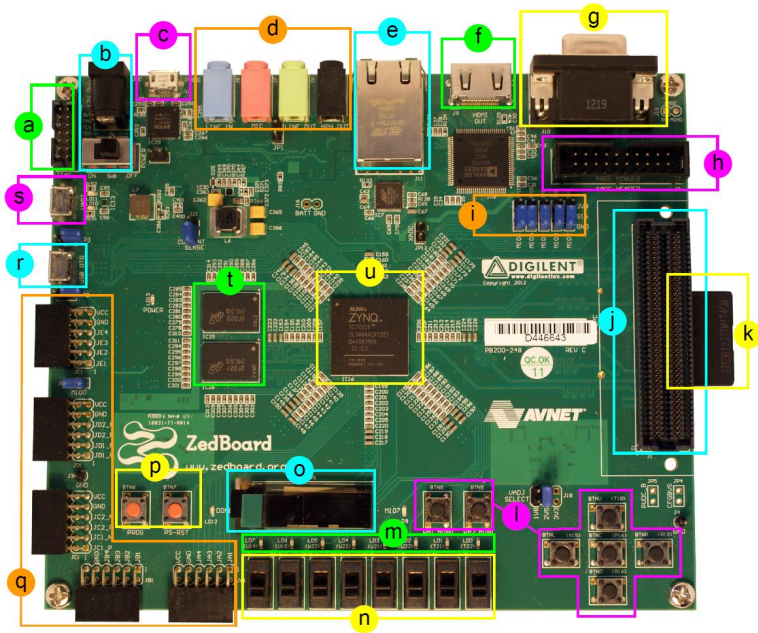
Our Board: Zynq ZedBoard

- ZedBoard: **Z**ynq **E**valuation and **D**evelopment **Board**



ZedBoard Layout and Interfaces

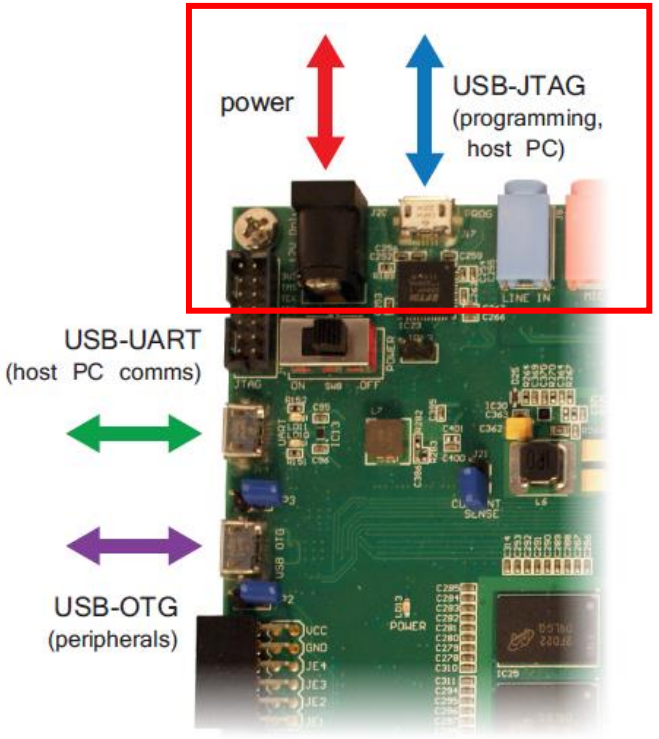
- ZedBoard features a ZC7Z020 Zynq device.
  - Artix-7 logic fabric, with a capacity of 13,300 logic slices, 220 DSP48E1s, and 140 BlockRAMs
  - DDR3 Memory, and Flash
  - Several peripheral interfaces



- a Xilinx JTAG connector
- b Power input and switch
- c USB-JTAG (programming)
- d Audio ports
- e Ethernet port
- f HDMI port (output)
- g VGA port
- h XADC header port
- i Configuration jumpers
- j FMC connector
- k SD card (underside)
- l User push buttons
- m LEDs
- n Switches
- o OLED display
- p Prog & reset push buttons
- q 5 x Pmod connector ports
- r USB-OTG peripheral port
- s USB-UART port
- t DDR3 memory
- u Zynq device (+ heatsink)

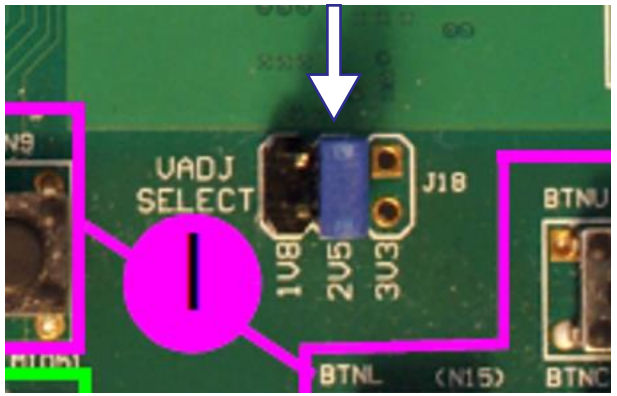
Hardware Setup for ZedBoard

- The ZedBoard must be connected to a power supply.
- By default, ZedBoard connects to the host PC for programming over USB-JTAG by micro-USB wire



- Check the **`Vadj Select`** is **`2v5`** at the I area (Near the buttons)

Vadj Select (J18)

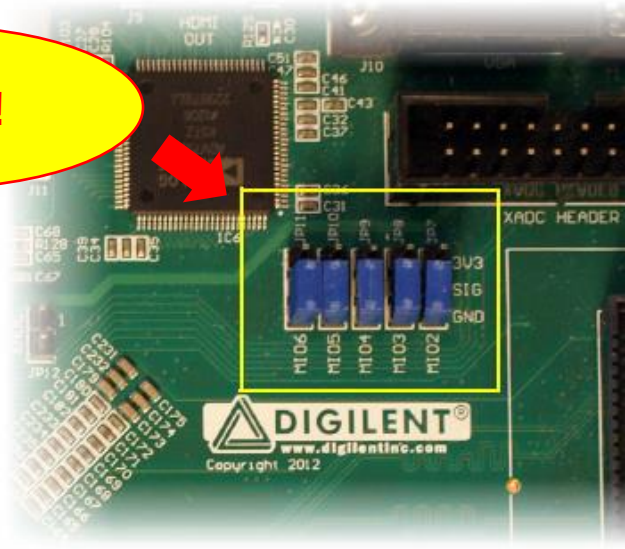


Programming the ZedBoard

- The ZedBoard user specifies the method of booting / programming via a set of jumper pins.
  - The middle three are for specifying programming source.

	MIO[6]	MIO[5]	MIO[4]	MIO[3]	MIO[2]
In Xilinx Technical Reference Manual...	Boot_ Mode[4]	Boot_ Mode[0]	Boot_ Mode[2]	Boot_ Mode[1]	Boot_ Mode[3]
JTAG Mode					
Cascaded JTAG[a]	-	-	-	-	-
Independent JTAG	-	-	-	-	1
Boot Device					
JTAG	-	0	0	0	-
Quad-SPI (flash)	-	1	0	0	-
SD Card[a]	-	1	1	0	-
PLL Mode					
PLL Used[a]	0	-	-	-	-
PLL Bypassed	1	-	-	-	-

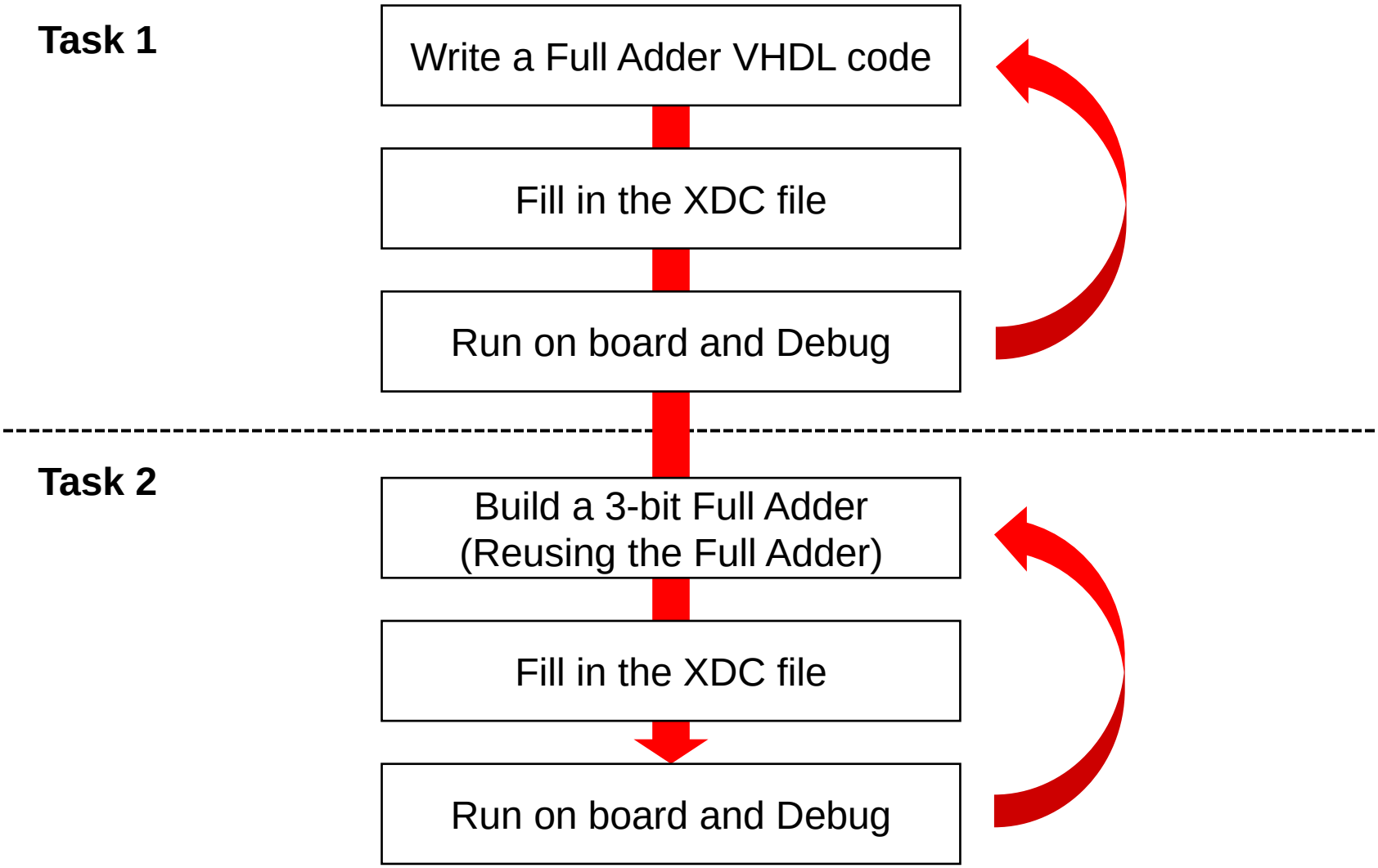
Cascaded: A single JTAG connection is used to interface to the debug access ports in both the PS and PL.



The PLL mode determines whether the process of configuring the device includes a phase of waiting for the PLL to lock

Lab02 - 3-bit Full Adder

Lab02 – 3-bit Adder Design Flow



Lab02 – Task 1

Procedures of each task

- **Procedures**

Called “lab02”

① Start the Vivado IDE and create a new project (see Lab01)
   - ➢ Remember to specify ZedBoard when creating a new project

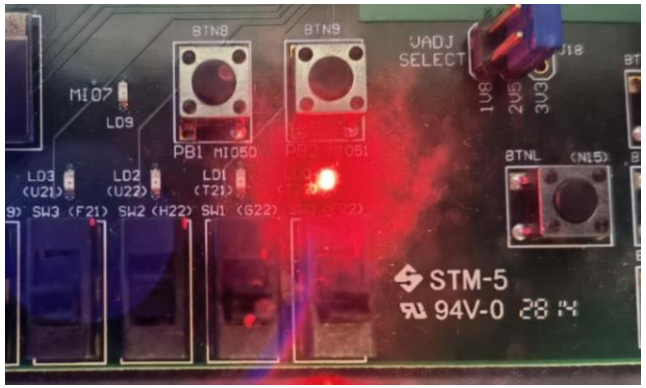
② Add a VHDL source file to your project (and implement)

③ Add constraints (i.e., an XDC file) to your project

④ Synthesis, implementation, and generate bitstream

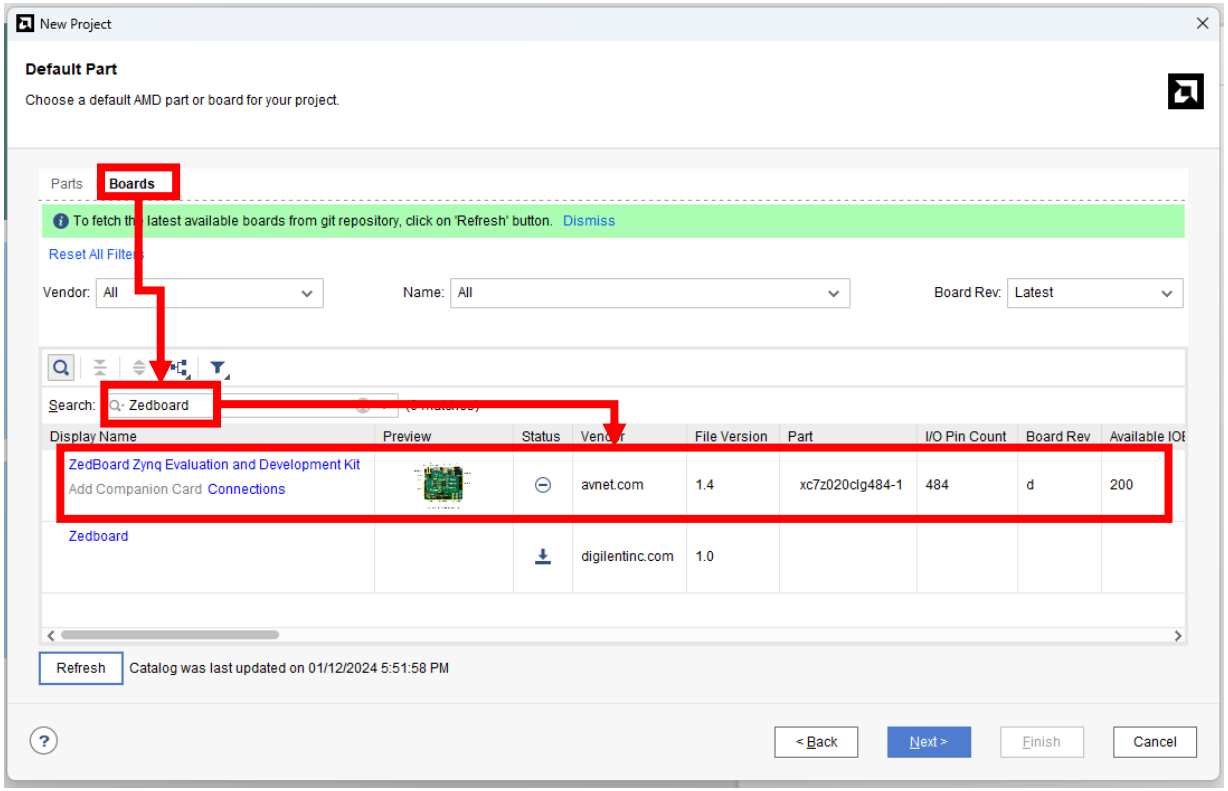
⑤ Download/Program your design to ZedBoard

⑥ Verify your design on ZedBoard



① Specifying ZedBoard in Vivado

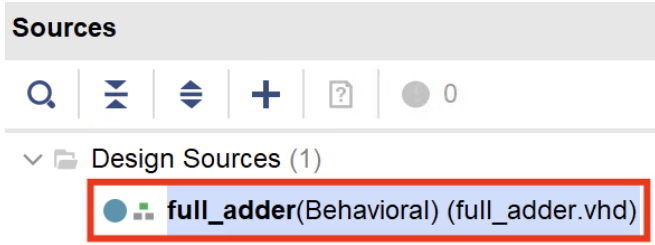
- **ZedBoard Zynq Evaluation and Development Kit**:
  - The design tools have knowledge of the specific facilities and peripheral connections of ZedBoard.



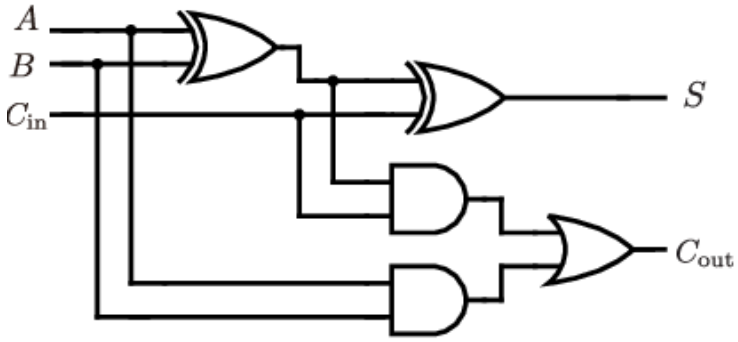
  - Refer to the appendix if ZedBoard is still not listed.

② Add a VHDL File & Implement (1/2)

- Create a design source called "**`full_adder`**"



- Implement the Full Adder



Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Style 1: Logic gates only

Style 2: when-else

② Add a VHDL File & Implement (2/2)

- Create a design source called "**`full_adder`**"

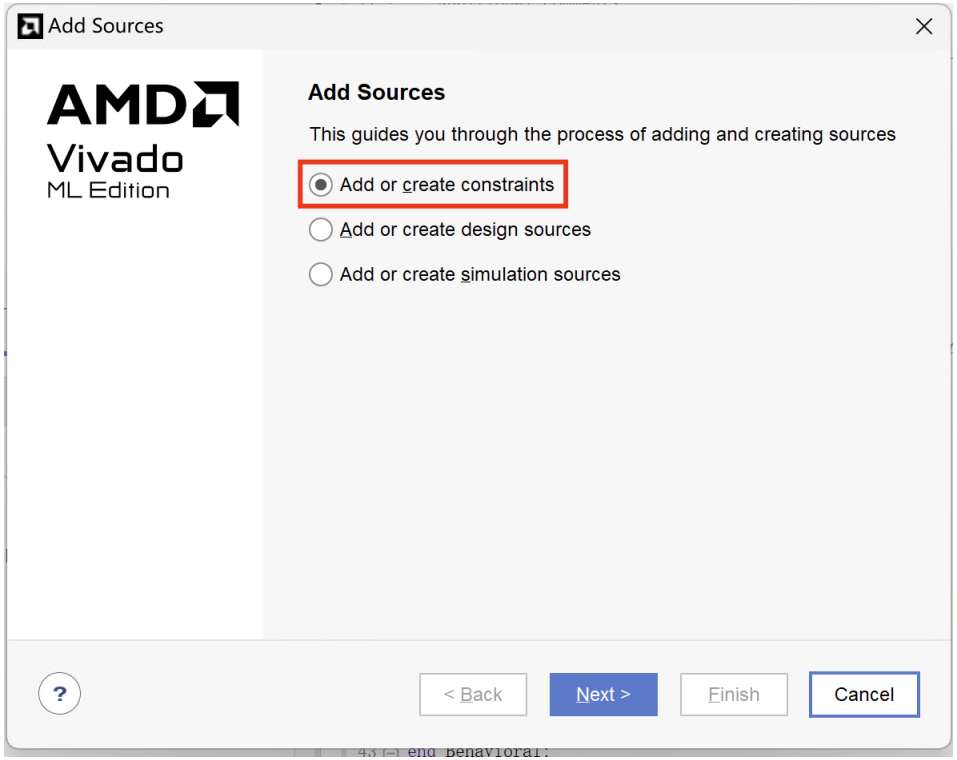
```
architecture Behavioral of
Full_adder is

begin
    Sum <= (A XOR B) XOR Cin;
    Cout <= ((A XOR B) AND Cin) OR
(A AND B);
end Behavioral;
```

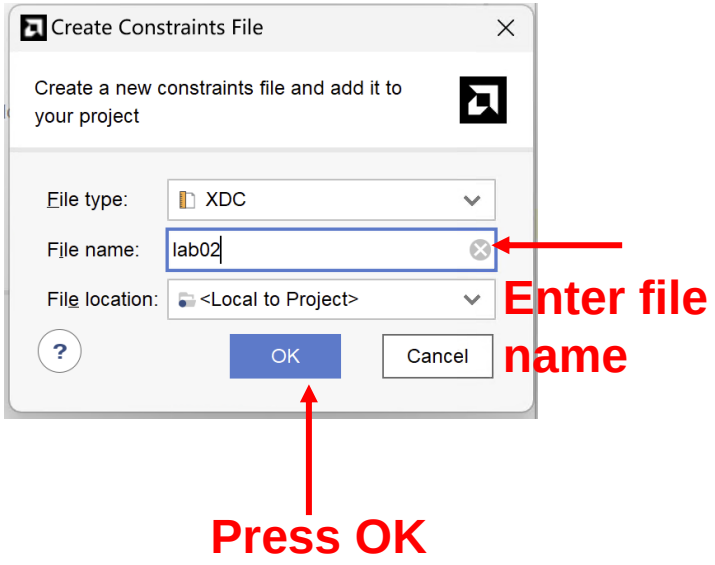
Style 1 reference code

You can copy this

③ Create XDC File (1/2)

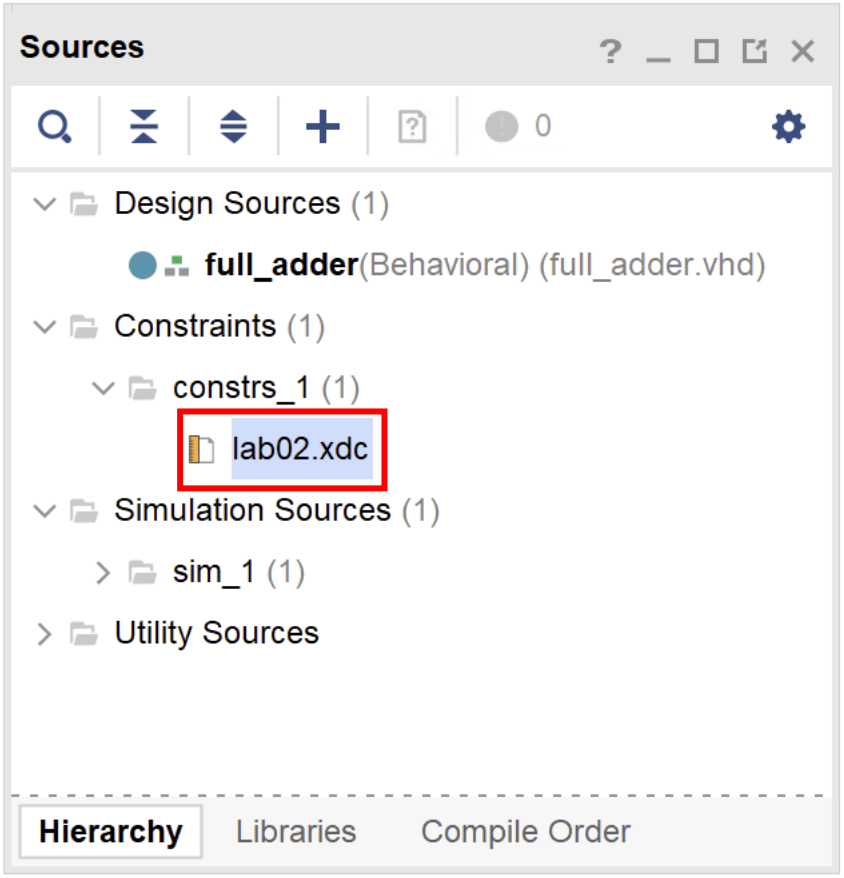
- Right Click “File->Add Sources->Add or create constrain->Create File->OK”
- Create a .xdc file called “lab02.xdc”

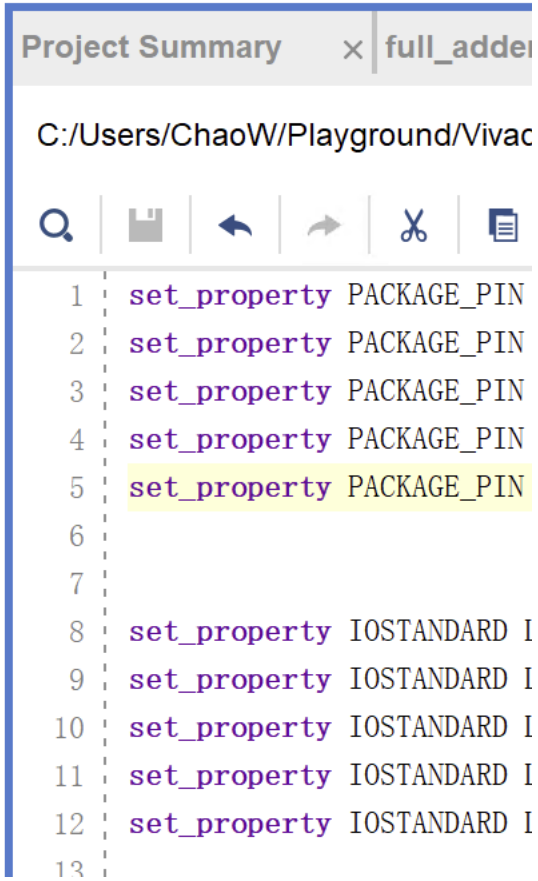




③ Create XDC File (2/2)

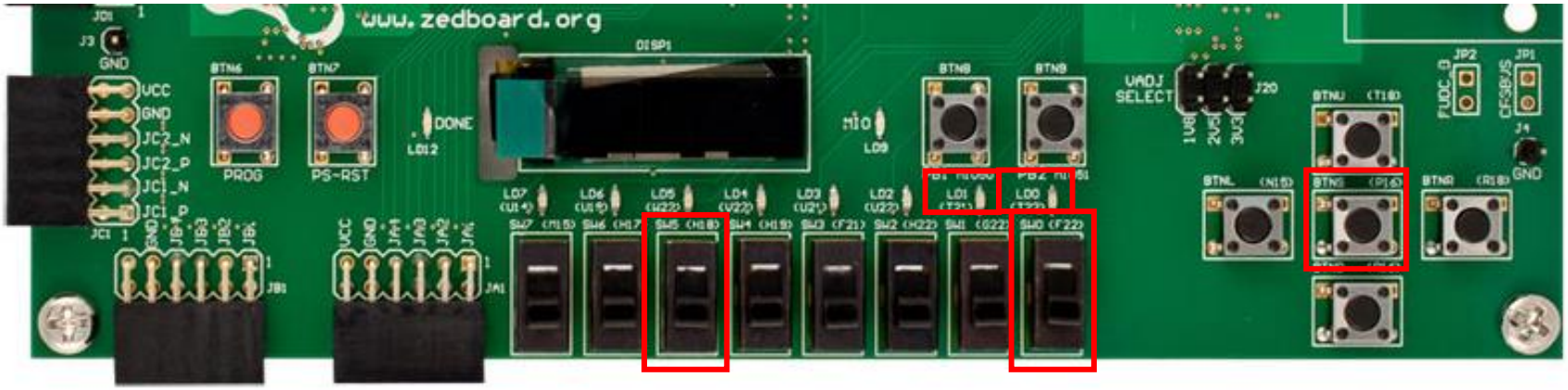
- Double Click “lab02.xdc” file
- Fill in the constrains with pin ID and voltage





.xdc File Specification

- Double Click “lab02.xdc” file
- Fill in the constrains with pin ID and voltage
  - A → SW5
  - B → SW0
  - Cin → BTNC
  - Sum → LED0
  - Cout → LED1



Board Specification

- Recall the syntax

```
set_property PACKAGE_PIN T22 [get_ports {C}]; # "LD0"
set_property PACKAGE_PIN F22 [get_ports {A}]; # "SW0"
set_property PACKAGE_PIN G22 [get_ports {B}]; # "SW1"

set_property IOSTANDARD LVCMOS33 [get_ports C]; # "LD0"
set_property IOSTANDARD LVCMOS25 [get_ports A]; # "SW0"
set_property IOSTANDARD LVCMOS25 [get_ports B]; # "SW1"
```

AND gate example in the Lecture

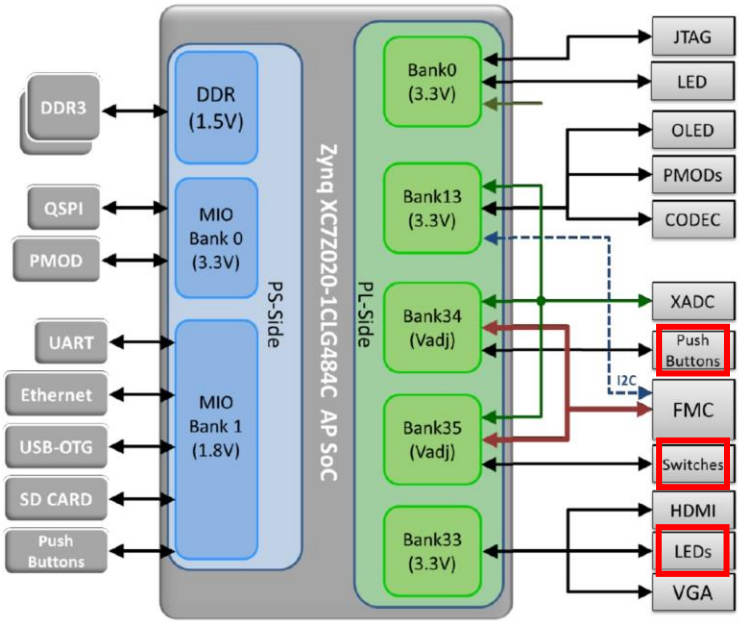
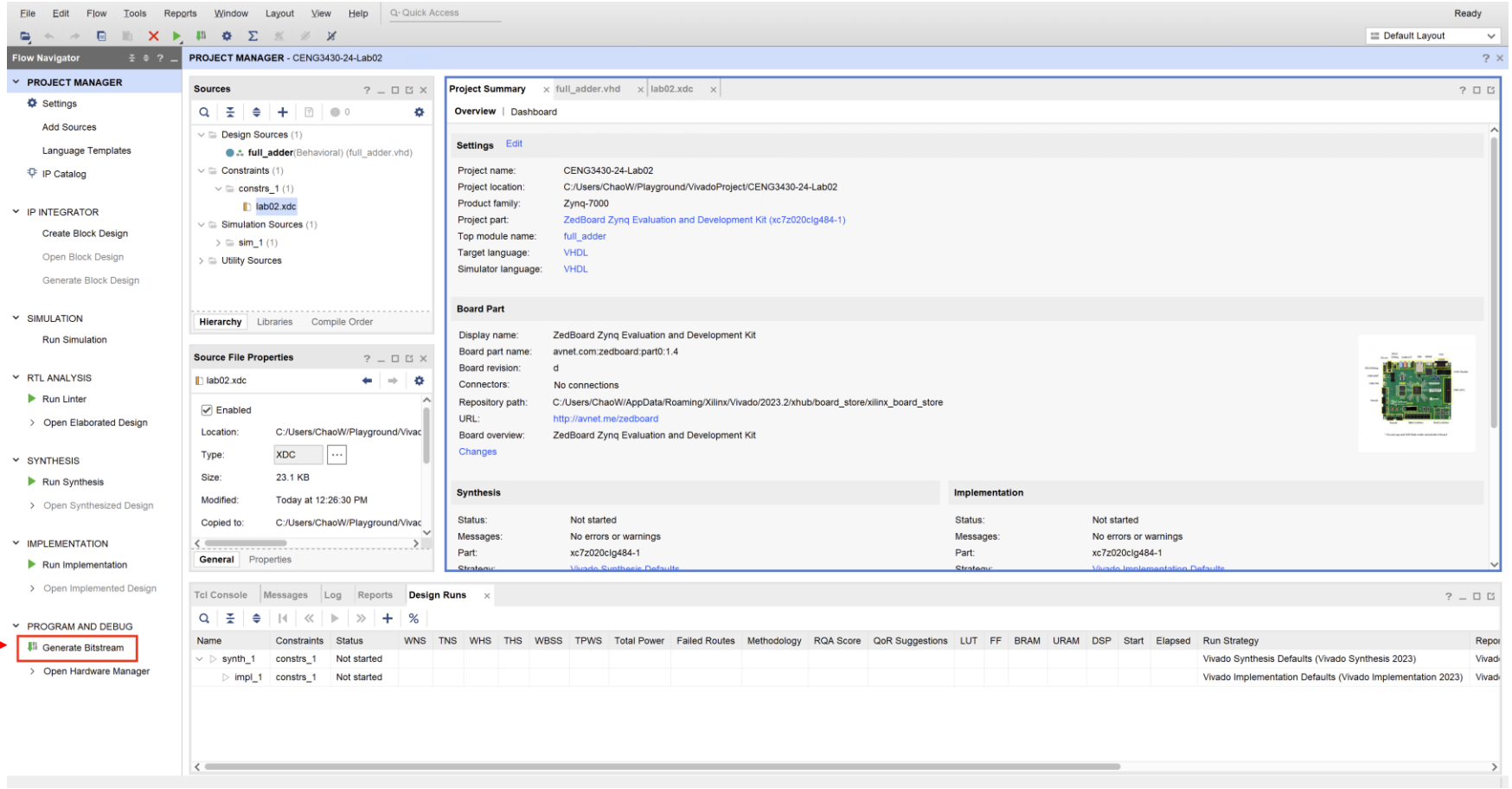


Figure 2 - Zynq Z7020 CLG484 Bank Assignments

Voltage Reference
(Details see Lec02)

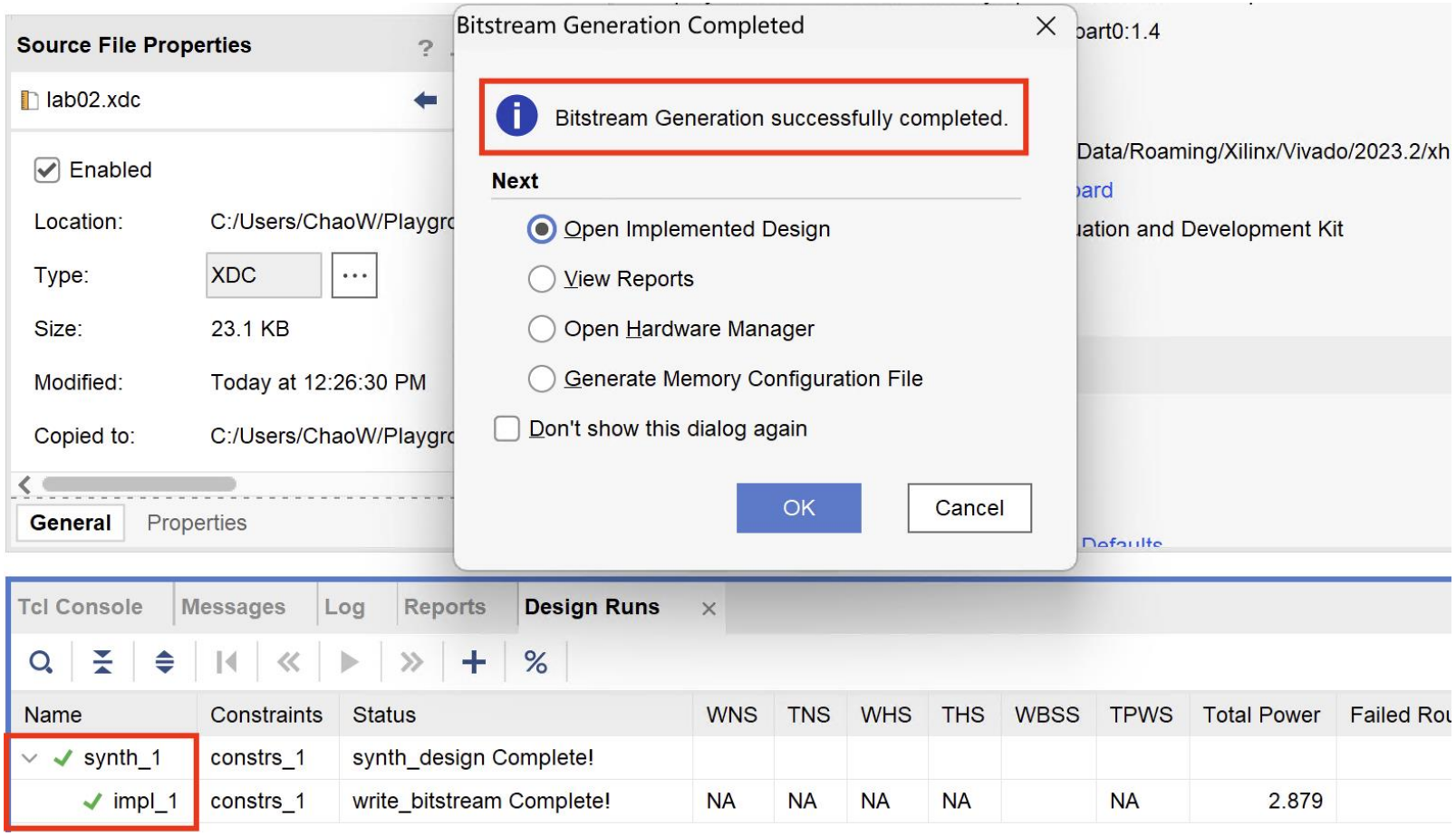
④ Synthesis, Implementation, Bitstream

- Click “Generate Bitstream”
  - Automatically run Synthesis, Implementation, and Generate Bitstream in sequence



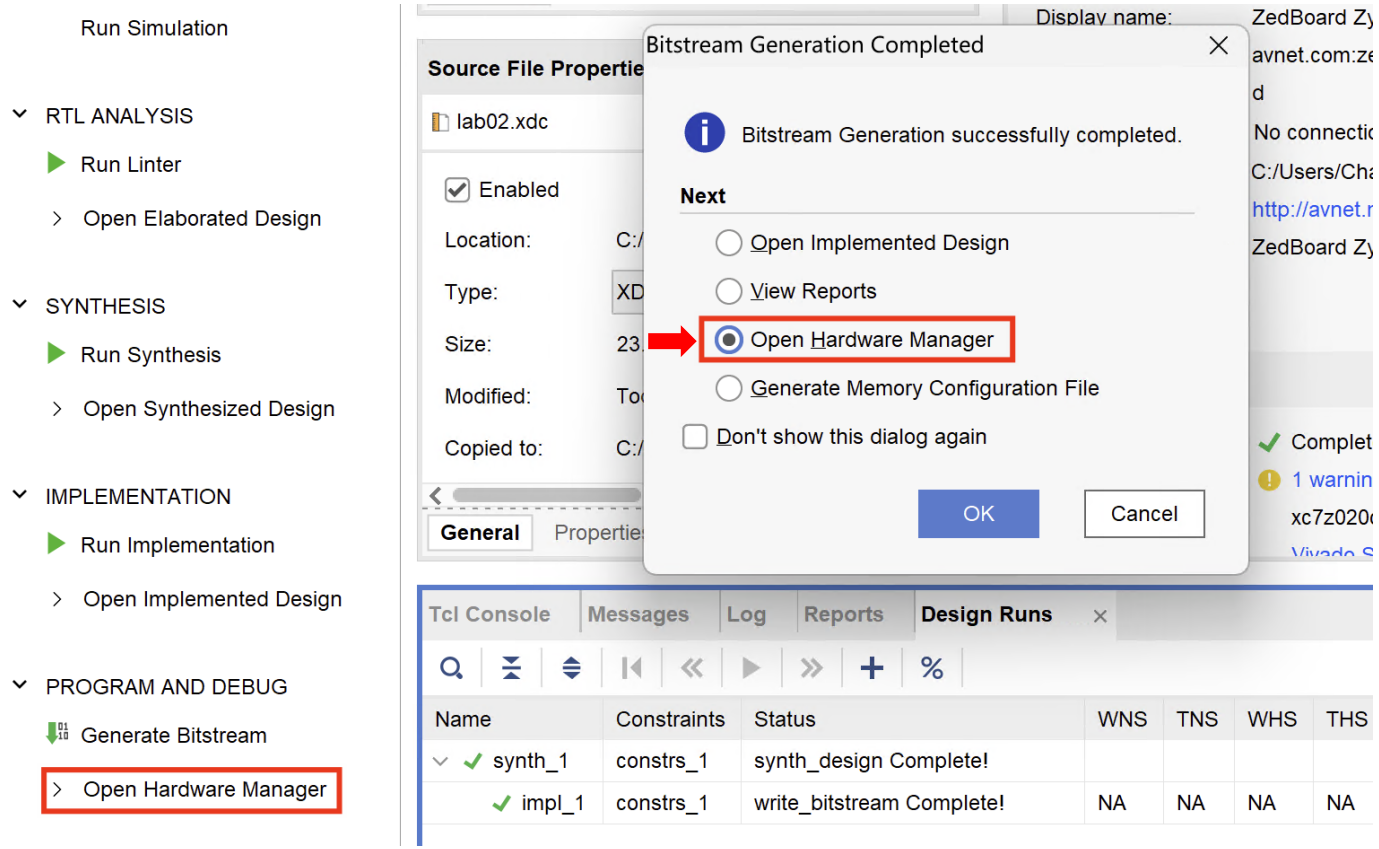
④ Synthesis, Implementation, Bitstream

- If there is no error, then you will see the successful message as shown on the picture:



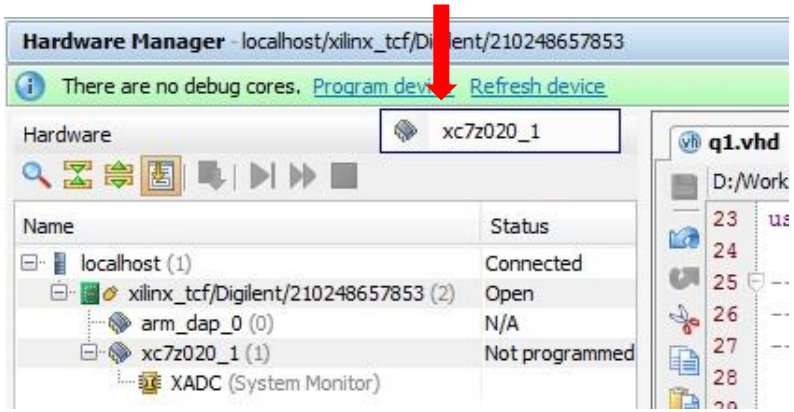
⑤ Download/Program Your Design

- Click “Open Hardware Manager” -> “OK”
- Make sure your FPGA is connected to PC



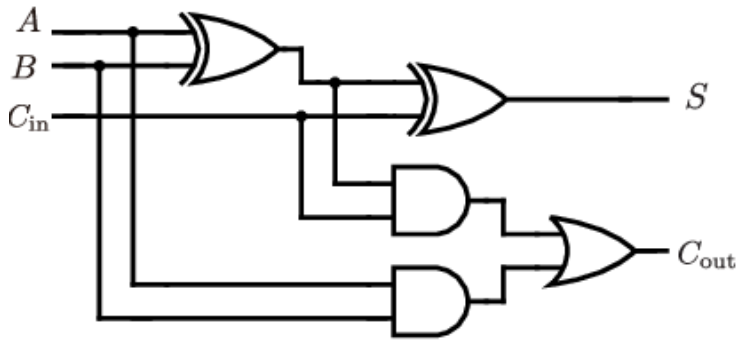
⑤ Download/Program Your Design

- If **Hardware Manager** reports **unconnected** press **Open target** and select **Auto Connect**.
- After the system successfully connected to the ZedBoard press **Program device** and select **xc7z020_1**. Then press **Program**.
- When the programming is successfully done the **blue LED** on the ZedBoard will turn on.



⑥ Verify Your Design on ZedBoard

- Verify the design using SW0, SW5, BTNC

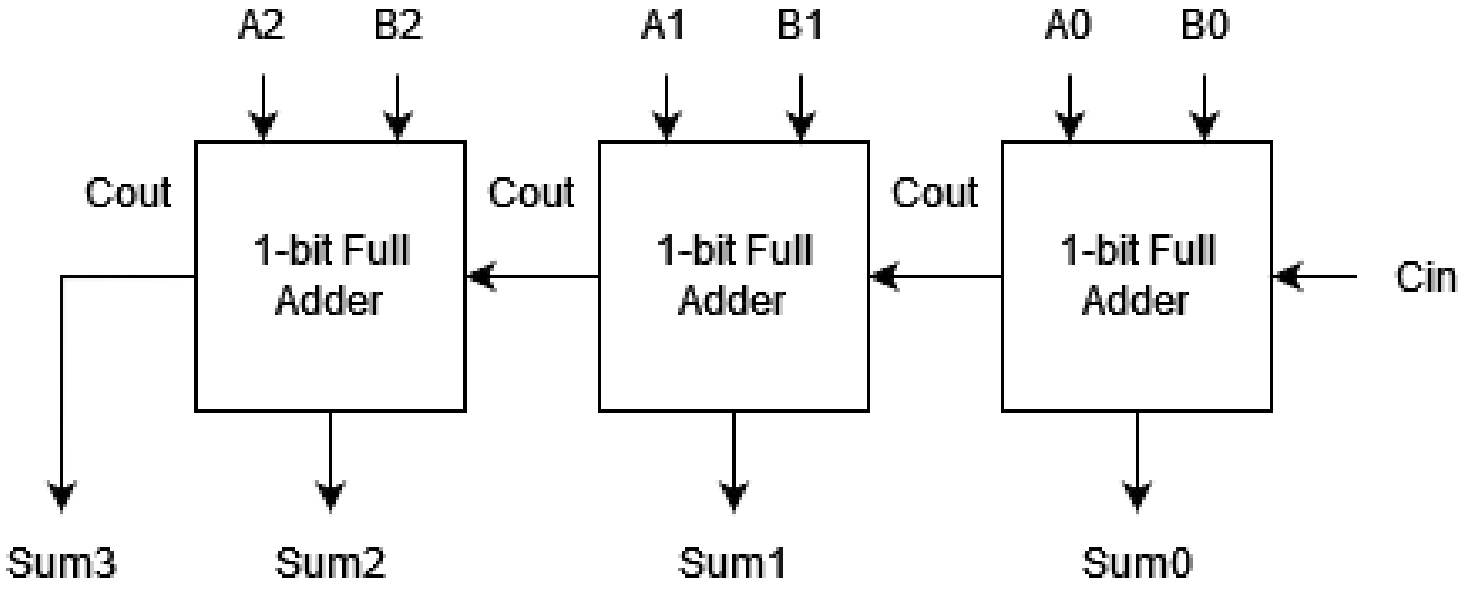


Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Lab02 – Task 2

Task 2

- **Same design procedure as Task 1**
- **Reusing the Full Adder to build a 3-bit Full Adder**



Step 1. Create a new design source

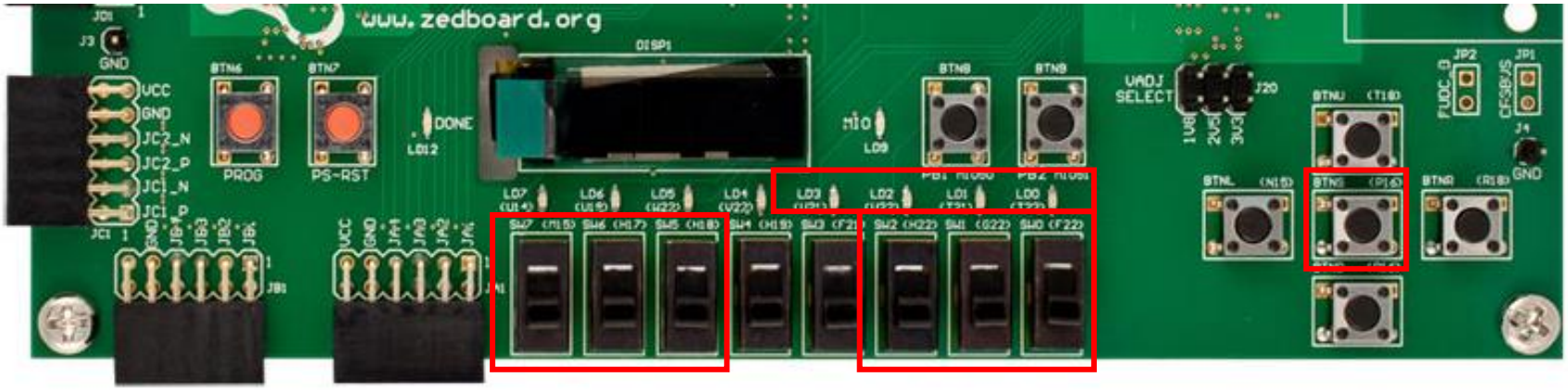
- Create a new design source called "**lab02**"
- The entity should be like this

```
entity lab02 is
    Port (
        A: IN STD_LOGIC_VECTOR(2 downto 0);
        B: IN STD_LOGIC_VECTOR(2 downto 0);
        Cin: IN STD_LOGIC;
        Sum: OUT STD_LOGIC_VECTOR(3 downto 0)
    );
end lab02;
```

- Must use component and port map to implement the 3-bit Full Adder by reusing the Full Adder module
  - The syntax of specifying a bit in std_logic_vector is **A(0)**

.xdc File Specification (1/2)

- Double Click “lab02.xdc” file
- Fill in the constrains with pin ID and voltage
  - A → SW5 - SW7 (A[0]: SW5, A[1]: SW6, …)
  - B → SW0 - SW2
  - Cin → BTNC
  - Sum → LED0 - LED3
- Specifying a pin in std_logic_vector is **`A[0]`**



.xdc File Specification (2/2)

- **Syntax Difference**
  - "A(0)" in .vhd file
  - "{A[0]}" in .xdc file

Wrong syntax in .xdc file may lead to compilation failure with no suggestions.

REALLY HARD TO DEBUG!!!

```
label1: Full_adder port map(A(0), B(0), Cin, C(0), Sum(0));
label2: Full_adder port map(A(1), B(1), C(0), C(1), Sum(1));
label3: Full_adder port map(A(2), B(2), C(1), Sum(2), Sum(3));
```

Lab2.vhd Sample

```
set_property PACKAGE_PIN H18 [get_ports {A[0]}]; #sw5
set_property PACKAGE_PIN H17 [get_ports {A[1]}]; #sw6
set_property PACKAGE_PIN M15 [get_ports {A[2]}]; #sw7
set_property PACKAGE_PIN F22 [get_ports {B[0]}]; #sw0
set_property PACKAGE_PIN G22 [get_ports {B[1]}]; #sw1
set_property PACKAGE_PIN H22 [get_ports {B[2]}]; #sw2
set_property PACKAGE_PIN P16 [get_ports {Cin}];  #BTNC
set_property PACKAGE_PIN T22 [get_ports {Sum[0]}]; #LED0(Cout0)
set_property PACKAGE_PIN T21 [get_ports {Sum[1]}]; #LED1(Cout1)
set_property PACKAGE_PIN U22 [get_ports {Sum[2]}]; #LED2(Cout2)
set_property PACKAGE_PIN U21 [get_ports {Sum[3]}]; #LED3(Cout3)

set_property IOSTANDARD LVCMOS25 [get_ports A[0]];
set_property IOSTANDARD LVCMOS25 [get_ports A[1]];
set_property IOSTANDARD LVCMOS25 [get_ports A[2]];
set_property IOSTANDARD LVCMOS25 [get_ports B[0]];
set_property IOSTANDARD LVCMOS25 [get_ports B[1]];
set_property IOSTANDARD LVCMOS25 [get_ports B[2]];
set_property IOSTANDARD LVCMOS25 [get_ports Cin];
set_property IOSTANDARD LVCMOS33 [get_ports Sum[0]];
set_property IOSTANDARD LVCMOS33 [get_ports Sum[1]];
set_property IOSTANDARD LVCMOS33 [get_ports Sum[2]];
set_property IOSTANDARD LVCMOS33 [get_ports Sum[3]];
```

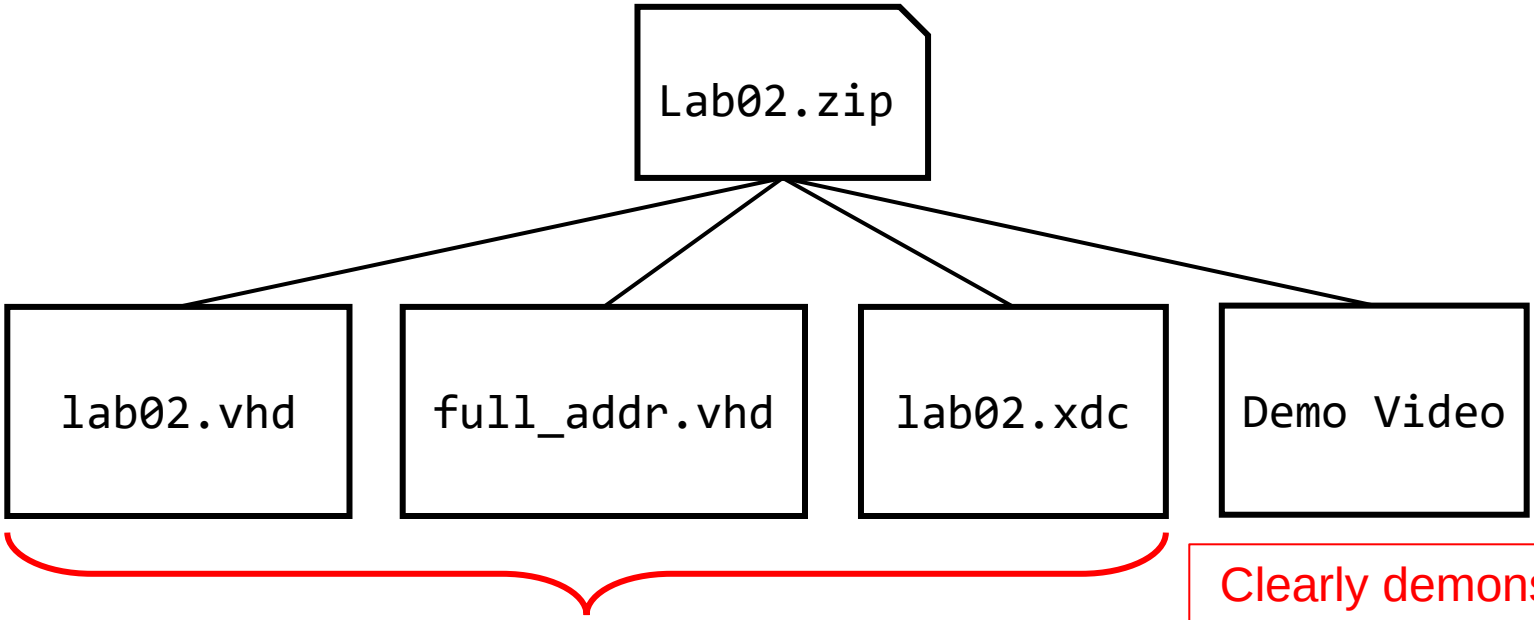
Lab2.xdc Sample

Demo Video Specification

- Since 3-bit Full Adder has 128 test cases, we only require you to demonstrate 5 test cases through demo video
- Each test case must present the result **with AND without the Cin='1'**

	Input		Expected Output	
Case	A	B	Cin=0	Cin=1
1	000	001	0001	0010
2	011	101	1000	1001
3	101	111	1100	1101
4	010	010	0100	0101
5	110	001	0111	1000

Lab02: Submission File Checklist



- **Submission Rule:**
  1. Submit the zip file following the above format to Blackboard
  2. Deadline: **12:30 on 05 Feb. 2025**
     - Late submission is NOT acceptable (unless otherwise approved)

香港中文大學
The Chinese University of Hong Kong

Thank You!

Appendix

- Is ZedBoard still not listed even after refreshing?
  - Plan A: utilize the script
    - Open the command prompt
    - Copy the following script to the command prompt.
    - Reopen Vivado, ZedBoard can be found.

```
REM set variables
SET https_proxy=proxy.cse.cuhk.edu.hk:8000
SET DOWNLOAD_URL=https://github.com/Digilent/vivado-boards/archive/master.zip
SET DOWNLOAD_PATH=C:\temp
SET UNZIP_PATH=C:\temp\
SET DEST_PATH=C:\Xilinx\Vivado\2023.2\data\boards\board_files
REM create folders
IF NOT EXIST %DOWNLOAD_PATH% (
mkdir %DOWNLOAD_PATH%
)
IF NOT EXIST %DEST_PATH% (
mkdir %DEST_PATH%
)
REM download board files
curl -Lk -o %DOWNLOAD_PATH%\master.zip %DOWNLOAD_URL%
REM unzip board files
IF EXIST %DOWNLOAD_PATH%\master.zip (
PowerShell -Command "Expand-Archive -Path '%DOWNLOAD_PATH%\master.zip' -DestinationPath '%UNZIP_PATH%'"
)
REM copy board files into vivado
xcopy /E /Y %UNZIP_PATH%\vivado-boards-master\new\board_files %DEST_PATH%
REM remove temp files
IF EXIST %UNZIP_PATH% (
rmdir /S /Q %UNZIP_PATH%
)
REM Success
```

  - Plan B: click here and follow the guide.