Xilinx Spartan 3E Starter Kit Tutorial

Hardware Description Languages - EE 310

Ahmet Can Mert

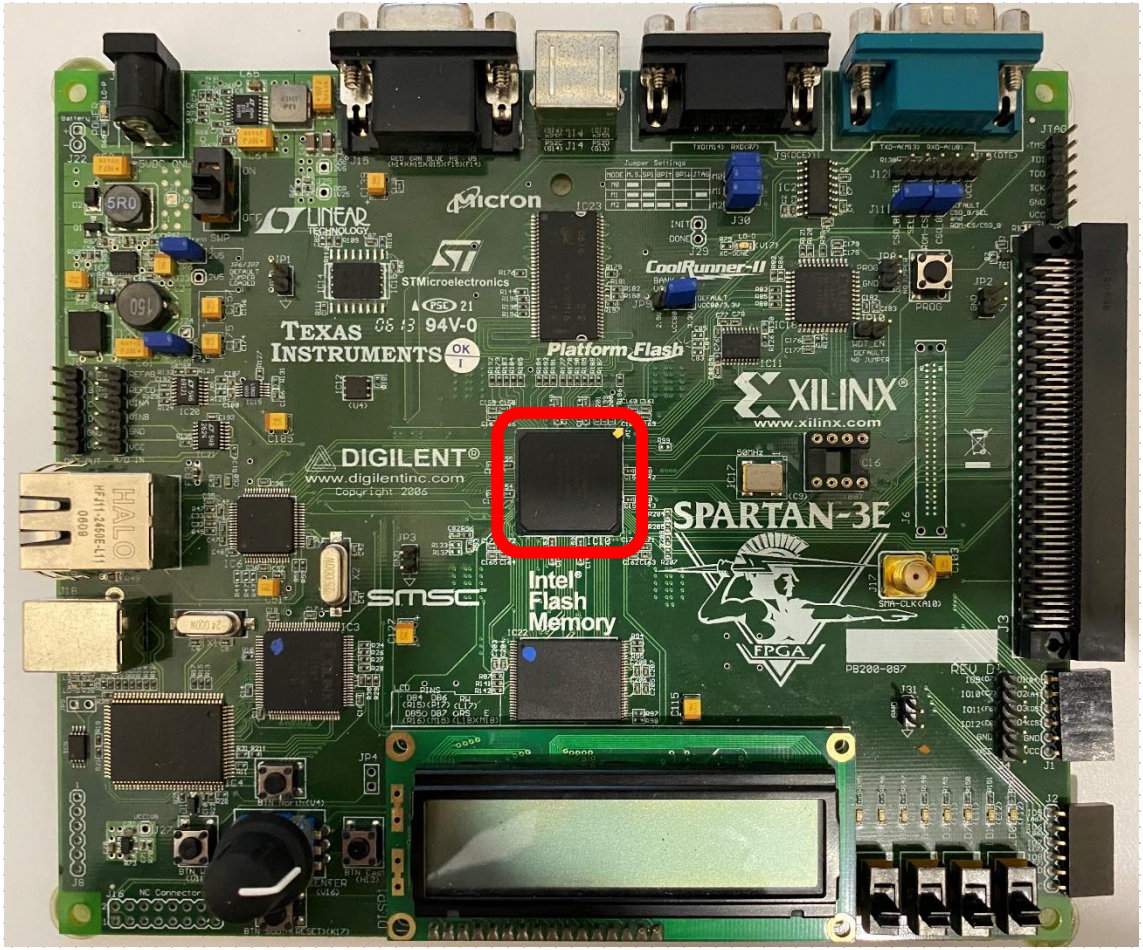
Sabanci University

Software & Hardware

- For Lab Assignments, you will use **Xilinx ISE Design Tool** to make design.
- You will implement your designs on Xilinx Spartan-3E Starter Kits. A Xilinx Spartan XC3S500E FG320 FPGA with speed grade 4 is placed on this board.
- During a design process, you will
  1. make your design on Xilinx ISE Design Tool
  2. simulate your design on ISim Simulator
  3. export your design to FPGA
     - 1. and 2. are already explained on Xilinx_ISE_Tutorial.pdf. In this tutorial, we will cover FPGA implementation.

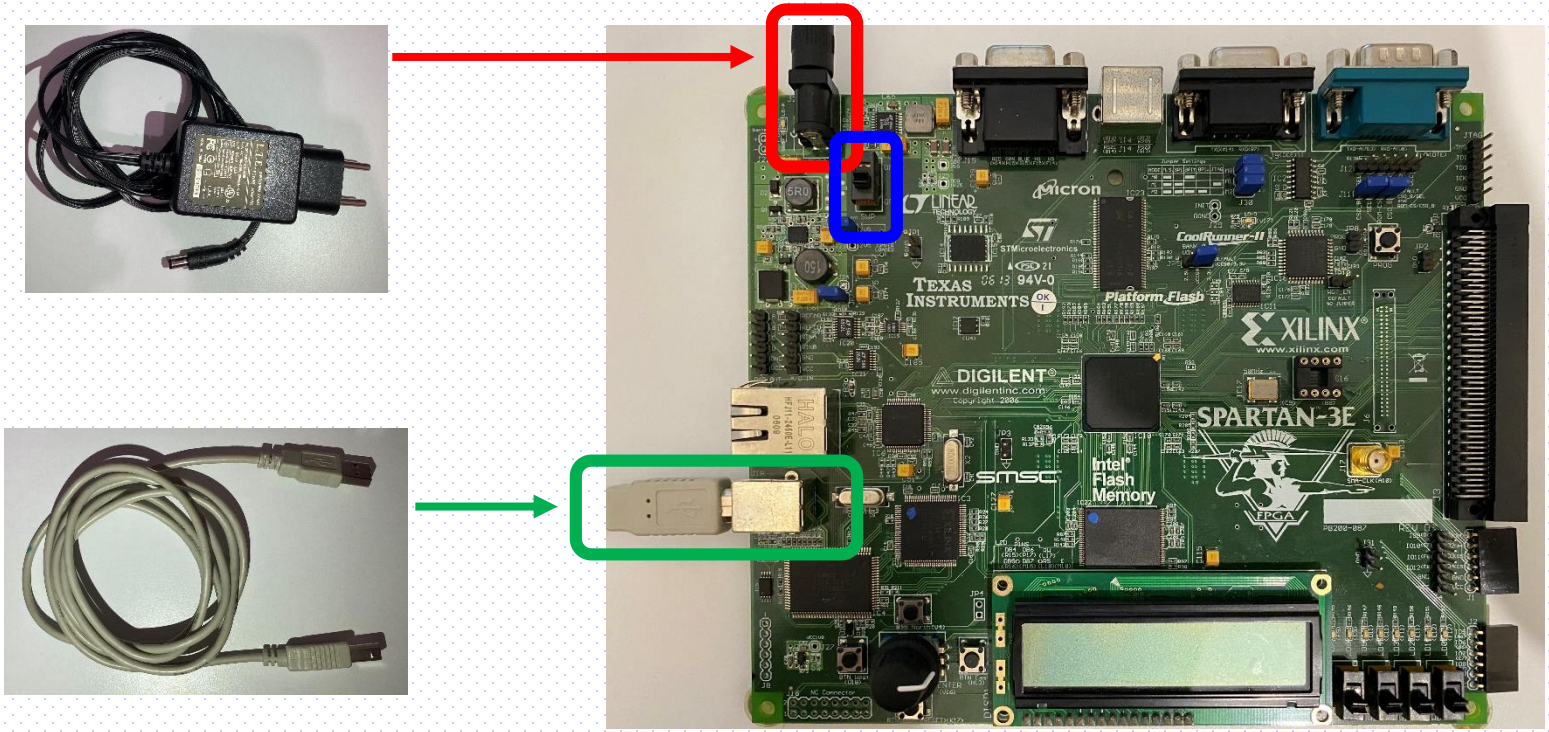
Spartan-3E Starter Kit

- Spartan-3E Starter Kit is an board which has a *Xilinx Spartan 3E* FPGA and different types of I/Os



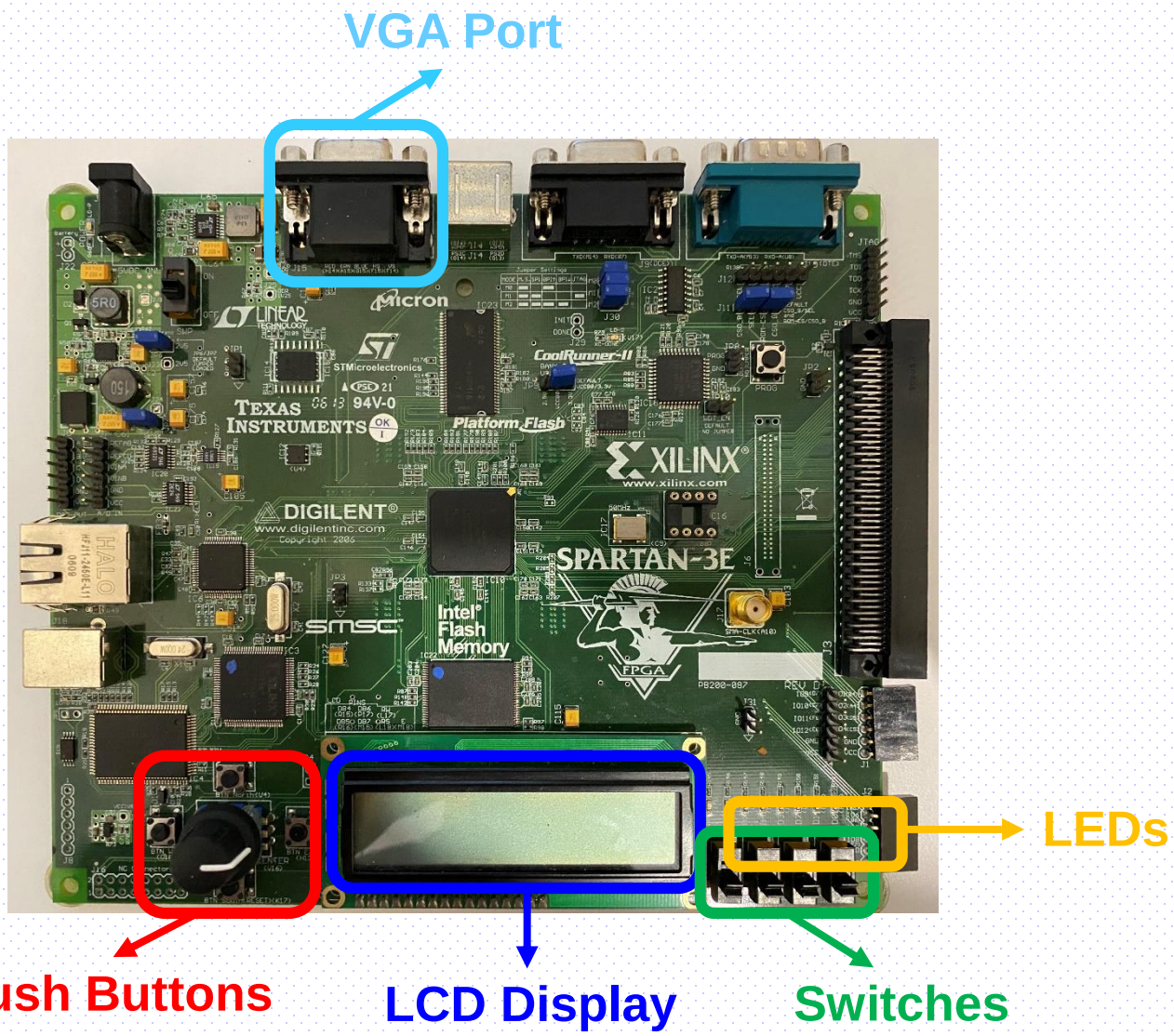
Spartan-3E Starter Kit

- The board is powered with an external adaptor and it is turned on/off with a switch. The board can be programmed via USB connection.



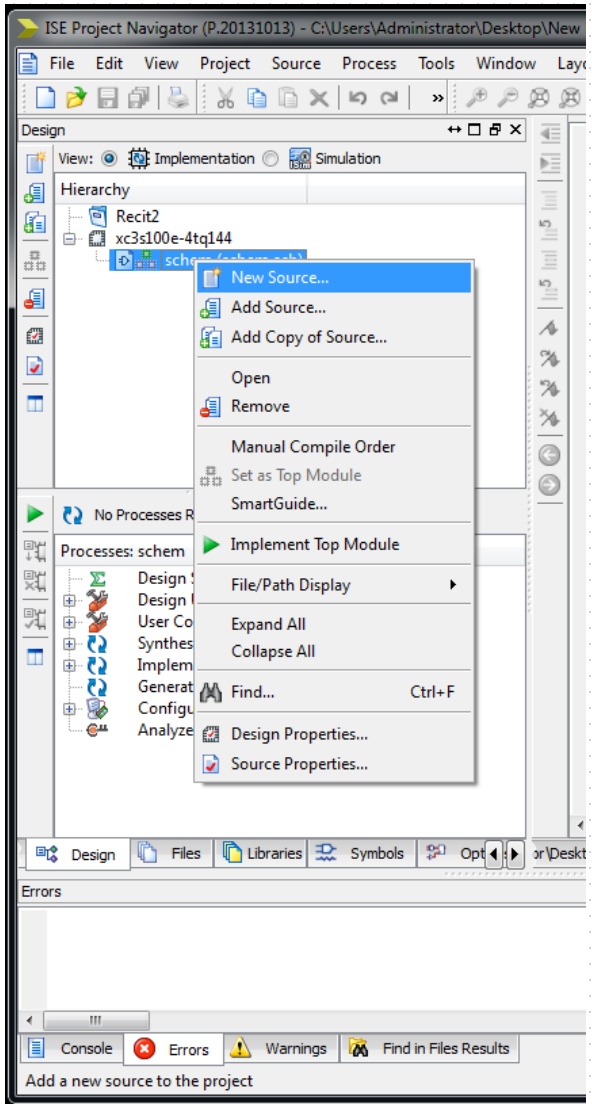
Spartan-3E Starter Kit

- I/O Ports:



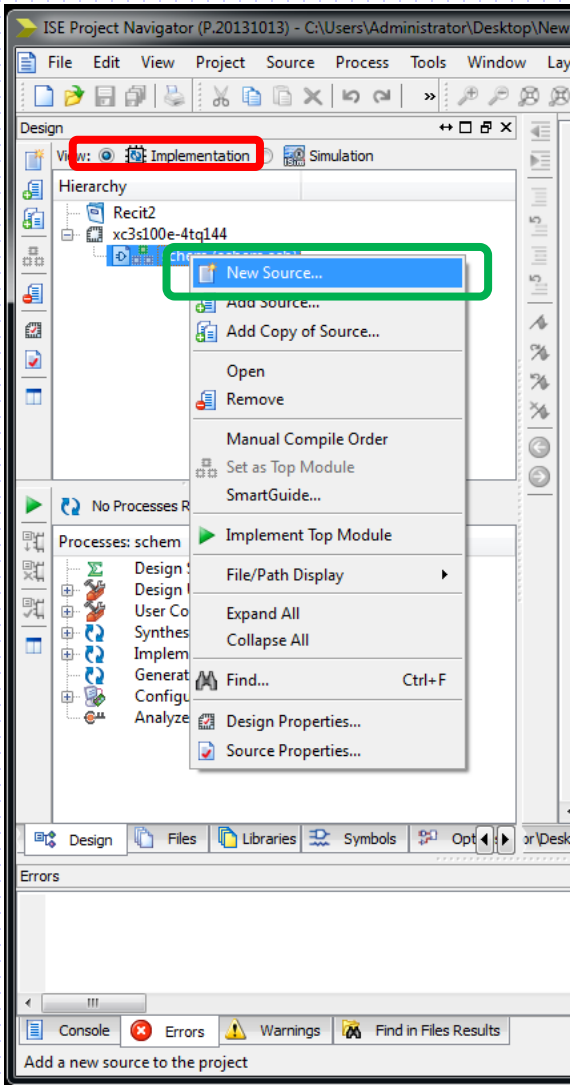
FPGA Implementation

- In order to run your design on FPGA, you need to define the hardware connections for your I/Os.
- The definitions are made on ***.ucf** file



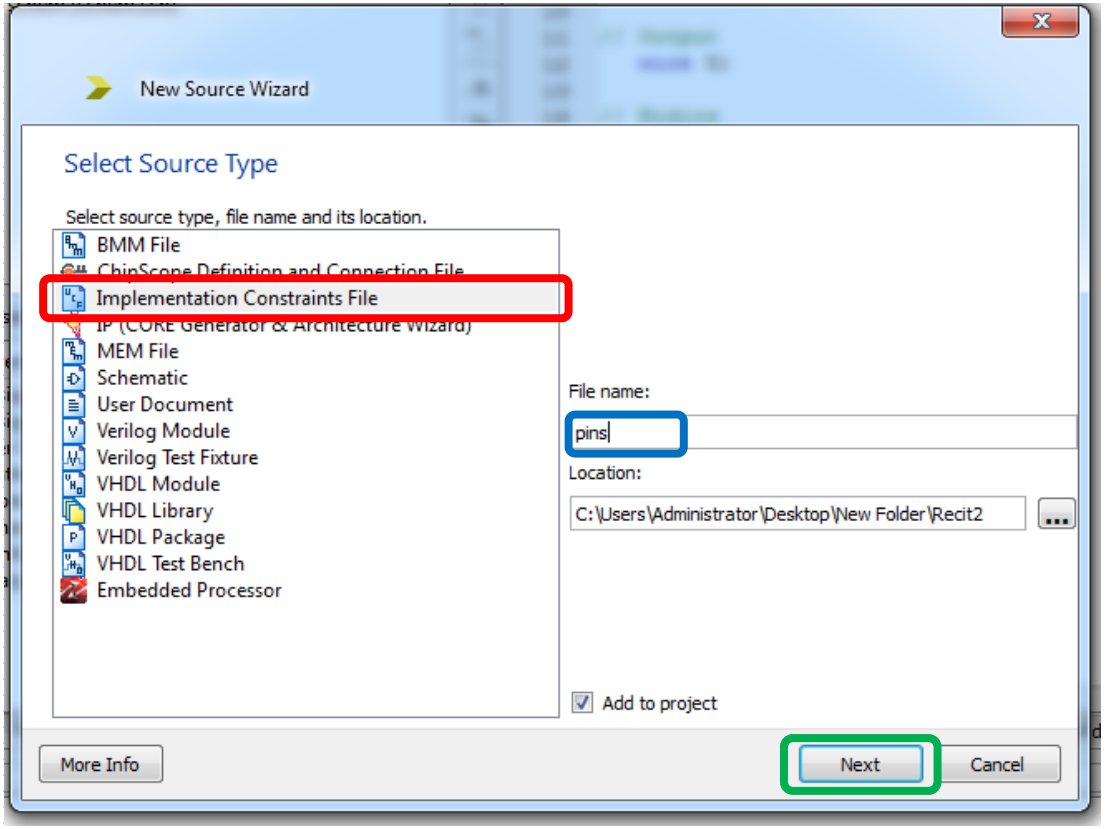
FPGA Implementation

- Select *Implementation* on Design panel
- Right-click on **.sch* file and select *New Source*



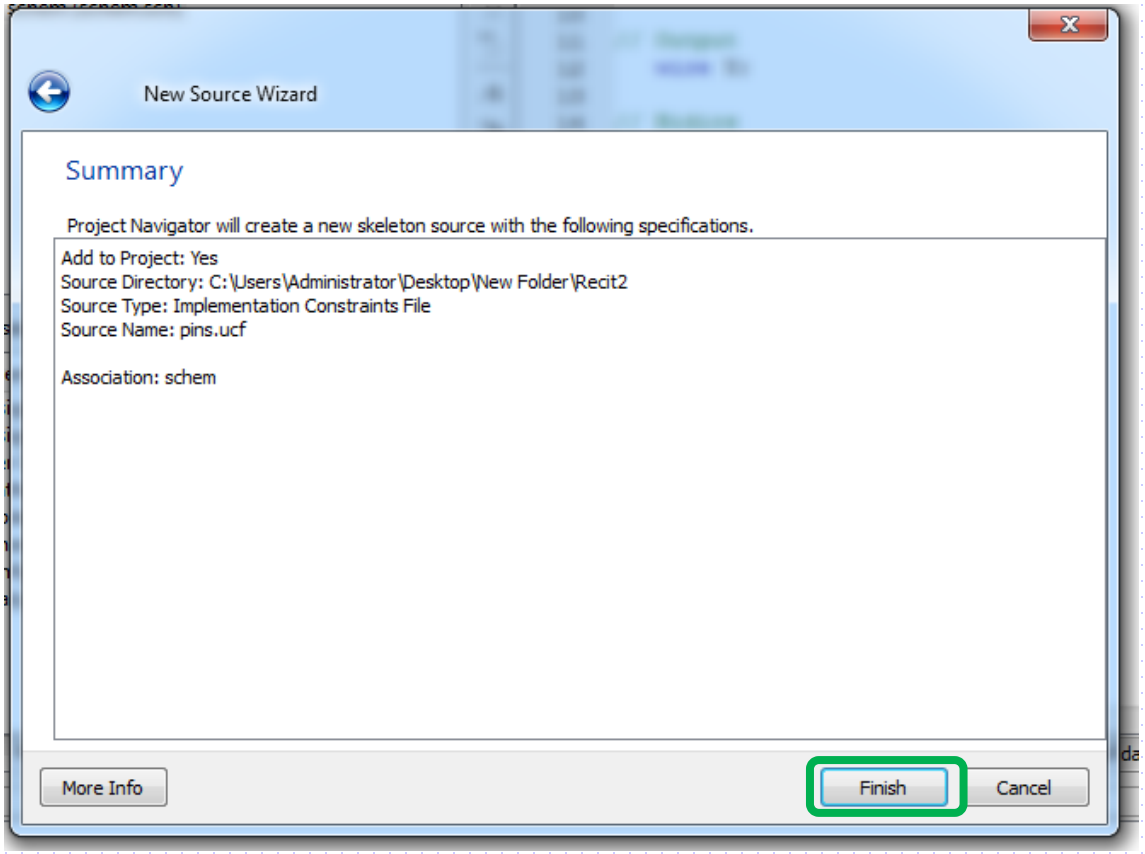
FPGA Implementation

- Select *Implementation Constraints File* as source type
- Give a name to file
- Click on *Next* button



FPGA Implementation

- Click on *Finish* button



FPGA Implementation

- There are different I/O ports and constraints we need to define in ***.ucf** file:
  - Clock source (on-board)
  - Clock period
  - Reset button
  - Push Buttons
  - Switches
  - LEDs
  - LCD Display
  - VGA Port

FPGA Implementation

- Syntax:

```
NET "SIGNAL_NAME" LOC = "FPGAPinLocation" | PROPERTIES;
```

  - SIGNAL_NAME: Name of input/output port
    - If it is a single bit register/wire:
      - e.g. `input start;` → `NET "start" LOC = "...`
    - If it is a bit of multi-bir register/wire:
      - e.g. `input data[3:0];` → `NET "data<0>" LOC = "...`
  - FPGAPinLocation: Pin Location of the port on FPGA
    - Specific code for each FPGA and port
  - PROPERTIES: Different properties/constraints for different ports

FPGA Implementation

- Clock source (on-board) in *.ucf file:
  - Syntax:

```
NET "CLK_NAME_IN_YOUR_DESIGN" LOC = "C9" | IOSTANDARD = LVCMOS33 ;
```

  - Example:

```
NET "clk" LOC = "C9" | IOSTANDARD = LVCMOS33 ;
```

- Clock period in *.ucf file:
  - Syntax:

```
NET "CLK_NAME_IN_YOUR_DESIGN" PERIOD = PERIOD HIGH DUTY_CYCLE%;
```

  - Example: (50 MHz clock with 50% duty cycle)

```
NET "clk" PERIOD = 20.0ns HIGH 50%;
```

FPGA Implementation

- Push button in *.ucf file:
  - Syntax:

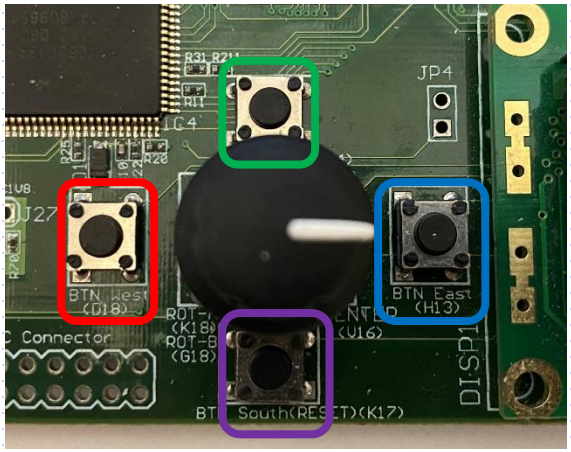
```
NET "SIGNAL_NAME_IN_YOUR_DESIGN" LOC ="FPGA_Pin_Location"| IOSTANDARD = LVTTL | PULLDOWN ;
```

  - Example:

```
NET "BTN0" LOC = "V4" | IOSTANDARD = LVTTL | PULLDOWN ;
```

  - Pin Locations

BTN West	D18
BTN North	V4
BTN East	H13
BTN South *	K17

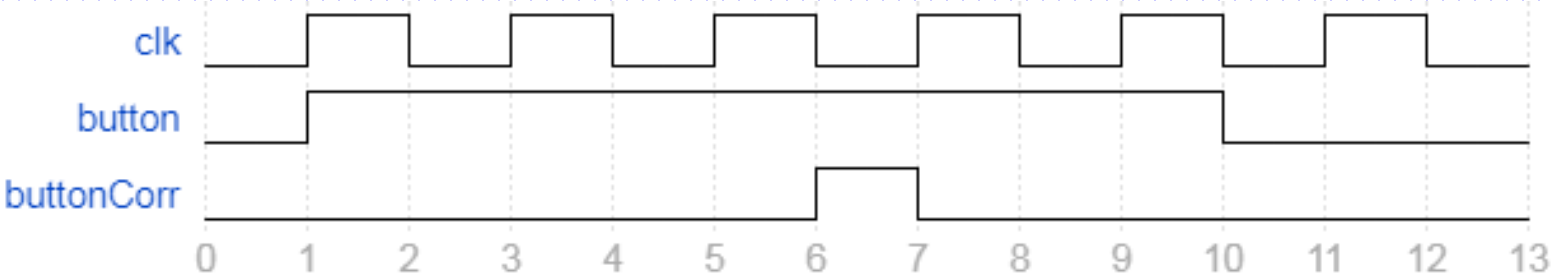


*Push button location K17 is used for Reset button

FPGA Implementation

- **debouncer** module
  - It is used to avoid that a single push on a button doesn't appear like multiple pushes
    - For this course, we will use it for filtering the button inputs
  - It is available under Resources→Lab Material→Modules
    - **debouncer.v**

```
debouncer db0(clk, button, reset, buttonCor);
```



FPGA Implementation

- Switches in *.ucf file:
  - Syntax:

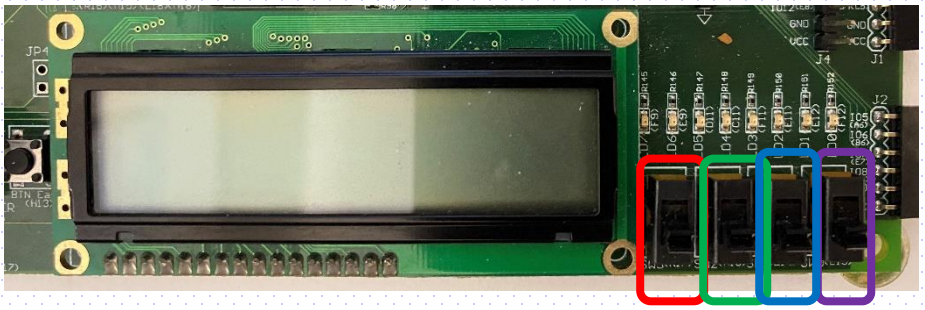
```
NET "SIGNAL_NAME_IN_YOUR_DESIGN" LOC ="FPGA_Pin_Location"| IOSTANDARD = LVTTL | PULLUP ;
```

  - Example:

```
NET "SW<0>" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;
```

  - Pin Locations

SW3	N17
SW2	H18
SW1	L14
SW0	L13



FPGA Implementation

- LEDs in *.ucf file:
  - Syntax:

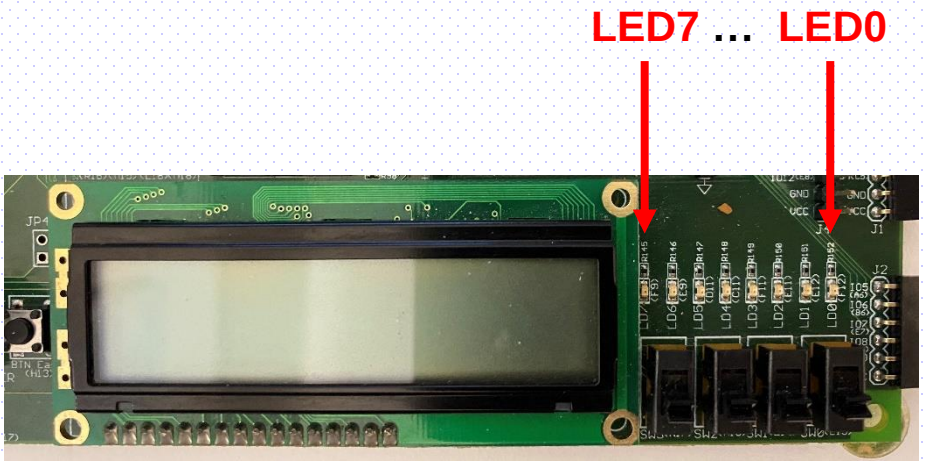
```
NET "SIGNAL_NAME_IN_DESIGN" LOC ="FPGA_Pin_Location"| IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
```

  - Example:

```
NET "LED<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
```

  - Pin Locations

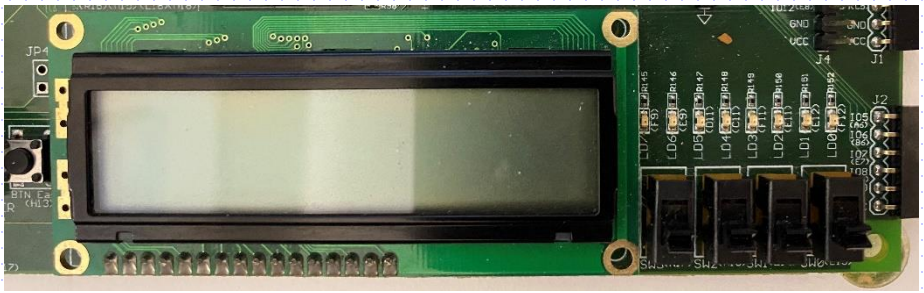
LED7	F9
LED6	E9
LED5	D11
LED4	C11
LED3	F11
LED2	E11
LED1	E12
LED0	F12



FPGA Implementation

- LCD Display in *.ucf file:
  - LCD Display on the board has 4 data and 3 control input signals
  - These ports are directly connected to the FPGA ports
  - Pin Locations

Data_Out<0>	R15
Data_Out<1>	R16
Data_Out<2>	P17
Data_Out<3>	M15
LCD_Control<0>	M18
LCD_Control<1>	L18
LCD_Control<2>	L17



FPGA Implementation

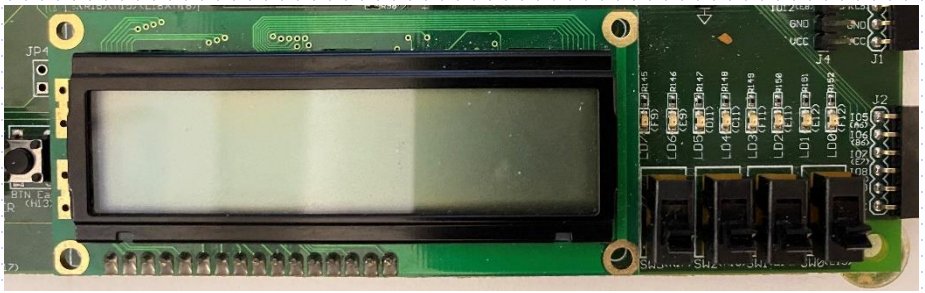
- LCD Display in *.ucf file:
  - Syntax:

```
NET "SIGNAL" LOC = "PIN" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
```

  - Example:

```
NET "Data_Out<0>" LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
```

Data_Out<0>	R15
Data_Out<1>	R16
Data_Out<2>	P17
Data_Out<3>	M15
LCD_Control<0>	M18
LCD_Control<1>	L18
LCD_Control<2>	L17



FPGA Implementation

- **LCDI** module
  - Since LCD Display on the board has a complex control protocol, we provide you a very simplified LCD interface
    - For this course, we will use it for displaying characters on LCDI
  - It is available under Resources→Lab Material→Modules
    - **LCDI.v**
  - It takes characters you want to display as input, generates necessary control signals and sends these signals to the LCD Display on the board

FPGA Implementation

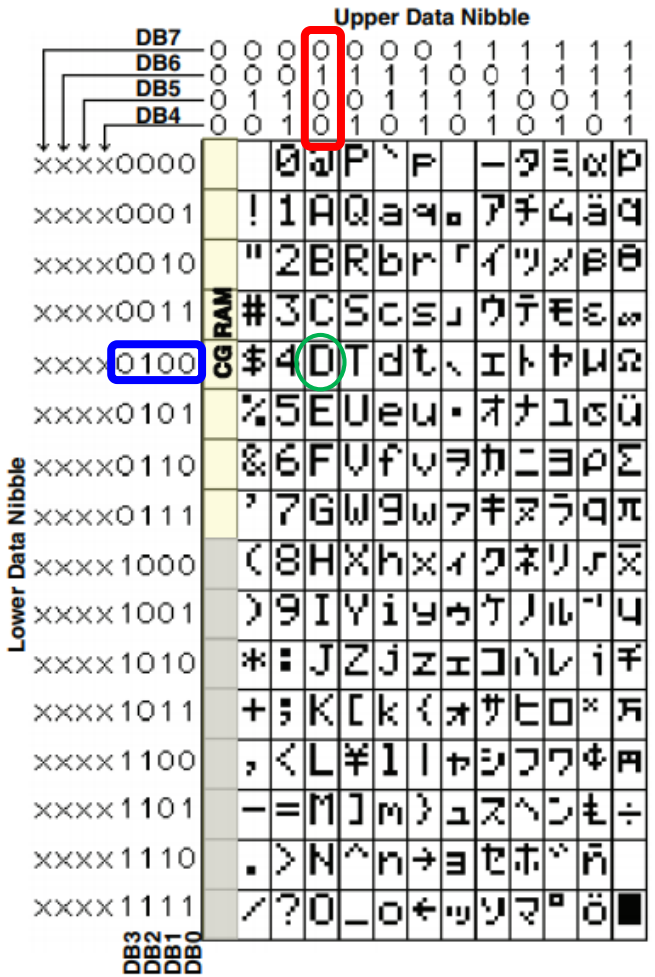
- **LCDI** module

```
LCDI d0(clk, First_line, Second_line, Data_Out, LCD_Control);
```

  - Input: 128-bit data for the first line
    - Each character is represented with 8 bits. Most-significant 8 bits of the input represents left-most digit.
  - Input: 128-bit data for the second line
    - Each character is represented with 8 bits. Most-significant 8 bits of the input represents left-most digit.
  - Output: 7-bit data (you do not care this, this is for LCD on board.)
  - Operation: It displays 32 (16 on the first line + 16 on the second line) characters on the display according to digit table

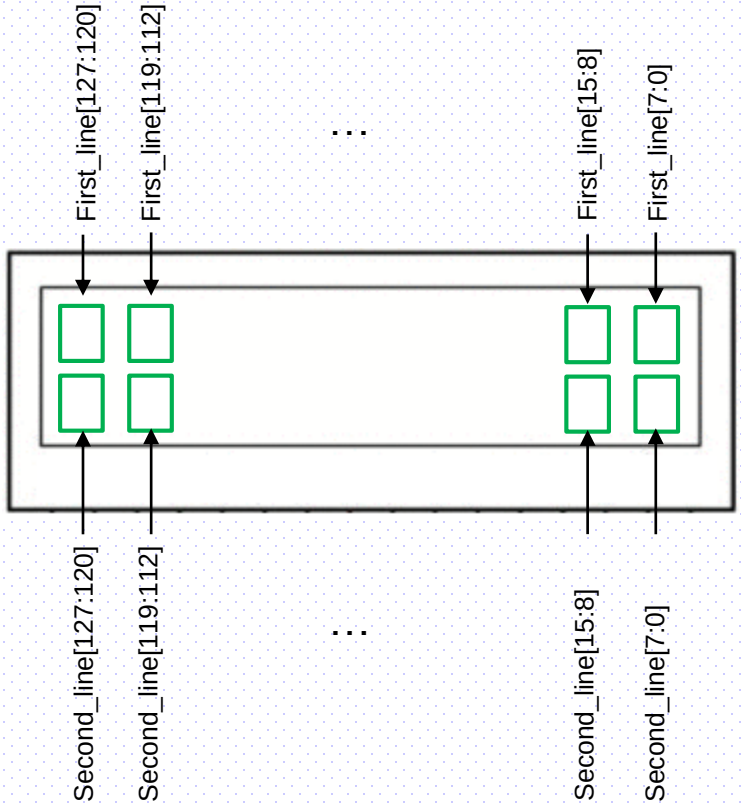
FPGA Implementation

- **LCDI** module
  - Digit table
  - Example:
    - D → 8'b01000100



FPGA Implementation

- **LCDI** module
  - Each character is represented with 8 bits according to digit table. Most-significant 8 bits of the line input represents left-most digit. Least-significant 8-bits of the line input represents right-most digit.



FPGA Implementation

- After you write *.ucf file, follow steps 16) – 25) of Xilinx_ISE_Tutorial.pdf