1. Creating New Project

1.1) Click on **Create New Project** (i.e,. Lecture_17)
Be sure there are no spaces in the folder name or path
(i.e., don't use "Lecture 17", but instead use "Lecture_17")
Otherwise, the tool chain will crash later on.
and click **Next**.

1.2) Choose Project Type as **RTL Project**. Leave the "Do not specify sources…" box unchecked and click **Next**. Don't add sources. **Next**. Don't add constraints. **Next**

1.3) select **Nexys Video** board. **Next**.

1.4) A summary of the new project design sources and target device is displayed. Click **Finish**.

2. Creating New Block Design

2.2) On the left you should see the Flow Navigator. Select **Create Block Design** under the IP Integrator. Keep the name as **design_1**, click **OK**.

2.3) Click the **Add IP button (plus sign)**. Search for "**Microblaze**" and double click on it to add the IP block to your empty design.

3. Adding Microblaze IP and Customization

3.1) click **Run Block Automation**

3.3) Change default settings in the block options
- Local Memory = **32KB**
- Cache Configuration = **16KB**
- Interrupt Controller = **checked (do not check for lecture 19 and lab#3)**

and click **OK**.

4. Customization of Clock Wizard IP Block

4.1) **Double click** on the **Clock Wizard**, clk_wiz_1, IP block.

4.2) Choose **sys clock** for **CLK_IN1**.
Choose **reset** for **EXT_RESET_IN**.

4.3) Select the **Output Clocks** tab.

4.4) **Check the box** next to **clk_out2**, then select clk_out2 output frequency as **200.000** (Mhz) and set **Reset Type** as **Active Low**. (scroll to bottom of window)

4.5) click **OK** to finish block automation of Clock Wizard.

5. Adding UART IP Block

5.1) Go to **Add IP (plus sign)** and search for "**UART**". Select the **AXI Uartlite** IP block.

6. Running Connection Automation for the First Time

6.1) Now select the **Run Connection Automation** from the Designer Assistance bar message prompt. Select **axi_uartlite_0**, **clk_wiz_1**, and **rst_clk_wiz_1_100M**. Do NOT select **microblaze_0**.

7. Adding and Customizing Memory Interface Generator IP Block

7.1) Click **Add IP (plus sign)** and search for "**Memory Interface Generator**", then double click the result.

7.2) click **Run Block Automation**. Click **OK**.

7.4) If you see this one error message [BD 41-1273]. You can ignore this. Click **OK** to dismiss this. If you have more than one error message, redo. Delete the MIG block from the schematic, and redo step 7.

Sometimes several MIG errors occur when the path name for your repo is too long. In this case, move your repo to a higher level directory.

8. Running Connection Automation for the Second Time

8.1) Now click on **Run Connection Automation**
8.2) **Select** only the **mig_7series_0** in the connection automation list.
- click on **sys_clk_i** and change **clock source** to **clock_out2 200MHz**.

Do not select **Microblaze_0** section.
Click **OK**.

10. Make DDR3 Signal External

10.1) The MIG block should be named **mig_7series_0**. Place your cursor on this symbol **||** next to the **DDR3+** port name. Your cursor will change to look like a pencil. **Right click** here and in the drop down list select **Make External**

11. Validate Design

11.1) Select **Validate Design** (**check box symbol or F6**).
If you get an error message, see the original tutorial.
11.2) Success? Click **OK**.

12. Creating HDL System Wrapper

12.1) click on ***sources*** *tab,* right click on ***design_1*** and select **Create HDL Wrapper**.
Let Vivado manage the wrapper.

13. Generating Bit File

13.1) In the **Flow Navigator panel** on the left, under **Program and Debug** select the **Generate Bitstream** option.
13.3) After the bitstream has been generated, a message prompt will pop-up on the screen. You don't have to open the Implemented Design for this demo. Just click on **Cancel**.

14. Exporting Hardware Design to SDK

14.1) On the top left corner of the window, from the tool bar click on **File** and select **Export Hardware**. Make sure the generated **bitstream** is included by **checking the box**.

15. Launching SDK

15.1) Go to **File** and select **Launch SDK** and click **OK**.

17. Creating New Application Project in SDK

17.1) Go to **File** in the main tool bar and select **New Application Project**. For the rest of this example, we assume you name your Project: **display_hello_world**. Since we only have one hardware design **design_1_wrapper_hw_platform_0** this will be our target hardware. Select
Create New under **Board Support Package**.
Click **Next**.

18. Selecting Hello World Application from available templates

18.1) Select **Hello World** under *Available Templates* on the left panel and click **Finish**.
18.2) **display_hello_world** is our main working source folder. This also contains an important file shown here which is the ***lscript.ld*****.** This is a Xilinx auto generated linker script file. **Double click** on this file to open.

19. Verify Linker Script File for Memory Region Mapping

19.1) In the linker script, take a look at the **Section to Memory Region Mapping** box. If you did the *Make DDR3 External* step then the target memory region **must** read **mig_7series_0**. Scroll down to check if this applies to all rows. If for any region it does not say **mig_7series_0**, then click on the row under the **Memory Region** column and select **mig_7series_0**.
19.2) Back in the *Project Explorer*, double click and open **helloworld.c** and make any edits you need to.

20. Programming FPGA with Bit File

20.1) Make sure that the Nexys Video board is turned on and connected to the host PC with the provided micro USB cable. Then click on the **Program FPGA** button to open the Program FPGA window. Make sure that the ***Hardware Platform*** is selected as **design_1_wrapper_hw_platform_0**.
In the software configuration box, under *ELF File to Initialize in Block RAM ()* column, the row option must read **bootloop**. If not, click on the row and select **bootloop**.
Now click on **Program**.

21. Run Configuration

21.1) From the *Project Explorer* panel, **right click** on the **display_hello_world** project folder. At the bottom of the drop down list, select **Run As** and then select **Run Configurations**.

The Run Configurations window is divided into two main sections. In the left panel, click on **Xilinx C/C++ application(GDB)** and then select **display_hello_world.elf**. *Note: In case you see **display_hello_world Debug** instead of display_hello_world.elf in this step, you can still run it without any issues.*

Now click on **Apply** and **Run**.

24. Use Tera Term Terminal Emulator

Note: SDK appears to no longer support UART messages in its console, so we will need to use an external terminal emulator like Tera Term. http://en.wikipedia.org/wiki/Tera_Term (http://en.wikipedia.org/wiki/Tera_Term) to know what Tera Term is. You can download and install Tera Term from this link http://ttssh2.sourceforge.jp/index.html.en

Establish a serial connection with the correct communication port inside Tera Term. Tera Term may find your COM port your USB is using automatically. If not you can find the COM port your USB is using my going to windows **Device Manager** and clicking on **Ports (COM & LPT)**. Mine is sometimes **COM3** and sometimes **COM5**. Typically the settings will be **8 Data Bits**, **No Parity Bit**, **1 Stop Bit**.

You should now see the Hello World message in Tera Term, and be able to type characters to microblaze using lec17.c