Setup Instructions for Hardware Assignments on Vivado

1 One-time GCL setup for Vivado

To know whether the given system is old or new, run the following command in the terminal after login into your GCL account:

```
§ lsb_release -a
```

If the Ubuntu version is 20.xx.x then you are using new system, otherwise you are working on the older machine.

Students who are using old systems in the DHD lab should follow the following instructions for **vivado** setup.

- Logon to the machine using your GCL account.
- In your home folder, open the .bashrc file and add the lines below (create a new file if .bashrc doesn't exist):

```
§ export XILINXD_LICENSE_FILE=2100@10.208.22.170:/extra/xilinxlic/Xilinx.lic
§ source /opt/Xilinx/14.7/ISE_DS/settings64.sh
§ source /opt/Xilinx/Vivado/2016.4/settings64.sh
```

- Alternatively, you can add the alias below and whenever you open a shell, run the command "enable_xilinx":

```
§ alias enable_xilinx="export XILINXD_LICENSE_FILE=2100@10.208.22.170:/extra/xilinxlic/Xilinx.lic;
  source /opt/Xilinx/14.7/ISE_DS/settings64.sh; source /opt/Xilinx/Vivado/2016.4/settings64.sh"
```

- After making changes to .bashrc, run "source .bashrc" in the terminal
- To run the required application, type **vivado** in the terminal.

Students working on the new systems don't need the license file and can directly log into their GCL account to start working on **vivado**.

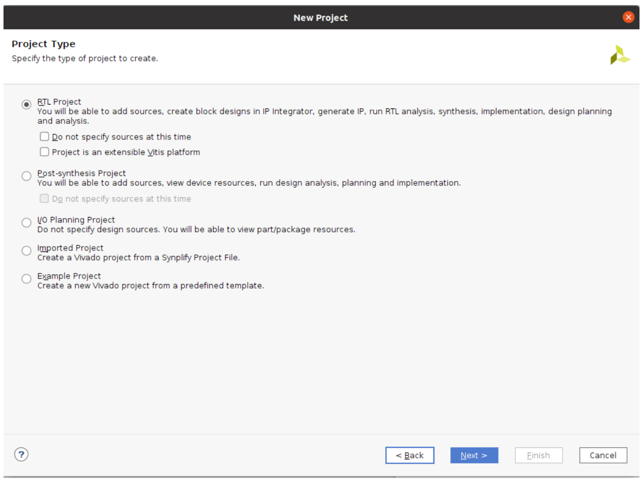
- In your home folder, open the .bashrc file and add the following line (create a new file if .bashrc doesn't exist)

```
§ source /opt/Xilinx/Vivado/2022.1/settings64.sh
```

2 Launching and creating a project in Vivado

1. To launch Vivado, type **vivado** in the terminal.
2. When the tool opens, click on **Create New project** and click Next.
3. Select the project directory and name.

4. In the next dialog box, select **RTL project.**



Project Type
Specify the type of project to create.

[✓ RTL Project]
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☐ Do not specify sources at this time
☐ Project is an extensible Vitis platform

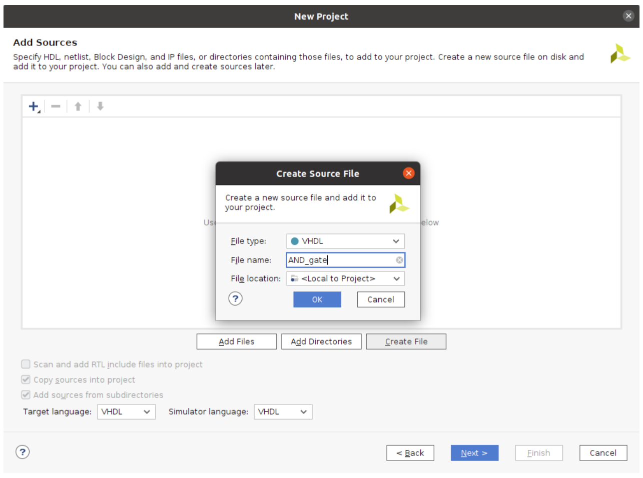
Post-synthesis Project
You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

Imported Project
Create a Vivado project from a Synplify Project File.

Example Project
Create a new Vivado project from a predefined template.

5. Create a new file named **AND_gate.** Select the file type as VHDL**.** Ensure that Target and simulator language is VHDL.



Add Sources
Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Create a new source file and add it to your project.

File type: VHDL
File name: AND_gate
File location: <Local to Project>

☐ Scan and add RTL include files into project
✓ Copy sources into project
✓ Add sources from subdirectories
Target language: VHDL Simulator language: VHDL

6. In the next dialog window, add the provided basys3.xdc file using "**Add Files**".
7. Select part number **xc7a35tcpg236-1** and click next. After this the project creation is done.
8. Under **design source**, double click **AND_gate** file to open it in the editor and add the following code for the basic AND gate.

```
-- AND GATE

entity AND_gate is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           c : out STD_LOGIC);
end AND_gate;
```

```
architecture Behavioral of AND_gate is

begin
    c <= a and b;
end Behavioral;

-- OR GATE
entity OR_gate is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           c : out STD_LOGIC);
end OR_gate;

architecture Behavioral of OR_gate is

begin
    c <= a or b;
end Behavioral;

-- MUX
-- Add your code for implementing a MUX using AND and OR gate
```

9. **Simulation of the design**: Now to simulate your design you need to make a test bench and test your code.
Test Benches are used to test the correctness of the Design Under Test or DUT. They can be used to provide inputs to the design and observe the outputs (for more information, please look at Slide 50-52 of the VHDL lecture Slides uploaded on Moodle).

   As a start, you can force the values of the inputs and observe their output in the waveform. Later in the tutorial, initialize inputs using memories as shown in Section 3.
Sample testbench is given for simulation of AND gate. You need to create testbench for MUX with all the input and output signals.
Note: Test Bench is only for simulation. Please do not run the implementation flow on it as introduced in the next steps.

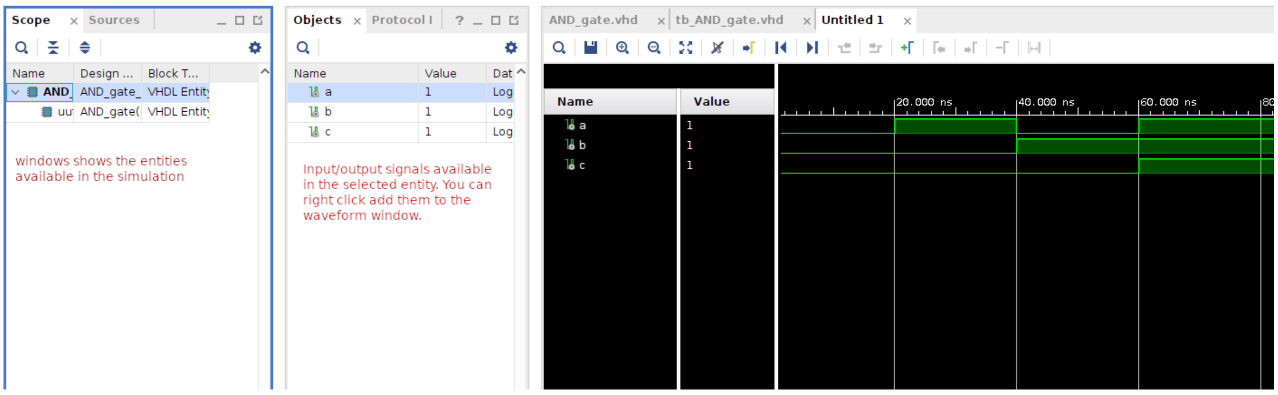
```
-- AND_gate_tb.vhd (testbench)

library ieee;
use ieee.std_logic_1164.all;

-- No ports for this TB so entity is empty
entity AND_gate_tb is
end AND_gate_tb;

architecture tb of AND_gate_tb is
    -- In this TB modeling Style, the test bench instantiates the DUT as a component
    -- and passes the inputs from a separate VHDL process via signals
    component AND_gate
      Port ( a : in STD_LOGIC;
             b : in STD_LOGIC;
             c : out STD_LOGIC);
    end component;
    signal a, b : std_logic;   -- inputs
    signal c : std_logic;   -- outputs
begin
    -- connecting testbench signals with AND_gate.vhd
    UUT : AND_gate port map (a => a, b => b, c => c);

    -- inputs
    -- 00 at 0 ns
    -- 01 at 20 ns, as b is 0 at 20 ns and a is changed to 1 at 20 ns
    -- 10 at 40 ns
    -- 11 at 60 ns
    a <= '0', '1' after 20 ns, '0' after 40 ns, '1' after 60 ns;
    b <= '0', '1' after 40 ns;
end tb ;
```



Name	Design ...	Block T...
AND	AND_gate_	VHDL Entity
uu	AND_gate(	VHDL Entity

windows shows the entities available in the simulation

Name	Value	Dat
a	1	Log
b	1	Log
c	1	Log

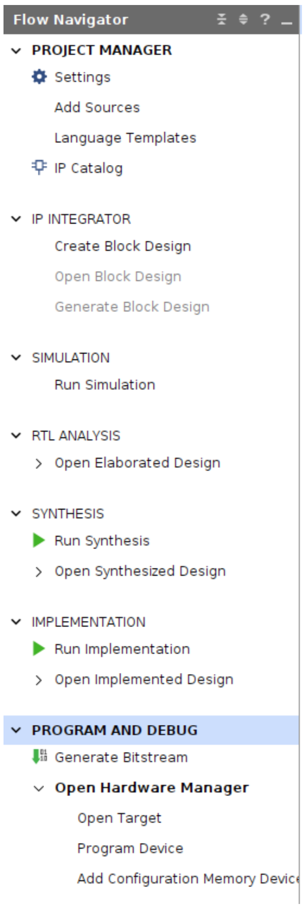
Input/output signals available in the selected entity. You can right click add them to the waveform window.

Name	Value
a	1
b	1
c	1

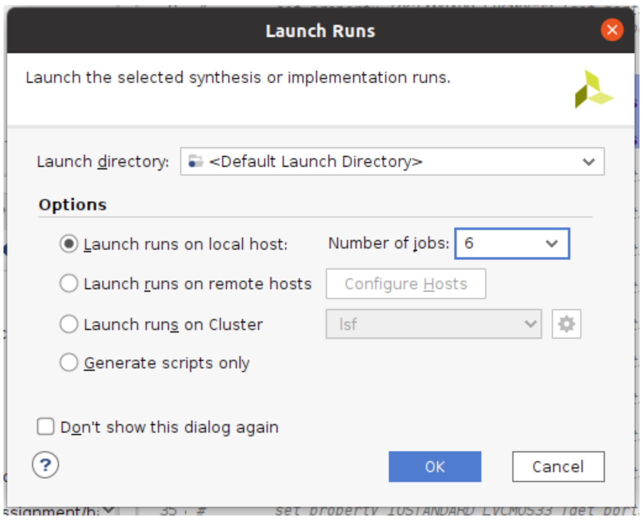
10. Next under the **Constraints section**, edit the **basys3.xdc** file to connect the switches and LED to the declared port. In this module, switches V17 and V16 act as inputs to the gate and LED U16 as the output of the gate.

```
set_property PACKAGE_PIN V17 [get_ports {a}]
        set_property IOSTANDARD LVCMOS33 [get_ports {a}]
set_property PACKAGE_PIN V16 [get_ports {b}]
        set_property IOSTANDARD LVCMOS33 [get_ports {b}]
....
....
....
set_property PACKAGE_PIN U16 [get_ports {c}]
        set_property IOSTANDARD LVCMOS33 [get_ports {c}]
....
....
```

11. Under the Flow Navigator, Click on **Run synthesis** then **Run implementation**. **Please open the synthesized design to analyze the output that is generated. Look at the resource utilization. Observe the number of LUTs used in your design.**



(a) Additional info: Number of jobs can be increased depending upon the available cores. This may improve the running time. For now, use the default value given in the dialog box.


Launch Runs

Launch the selected synthesis or implementation runs.

Launch directory: <Default Launch Directory>

Options

(●) Launch runs on local host: Number of jobs: 6

( ) Launch runs on remote hosts [Configure Hosts]

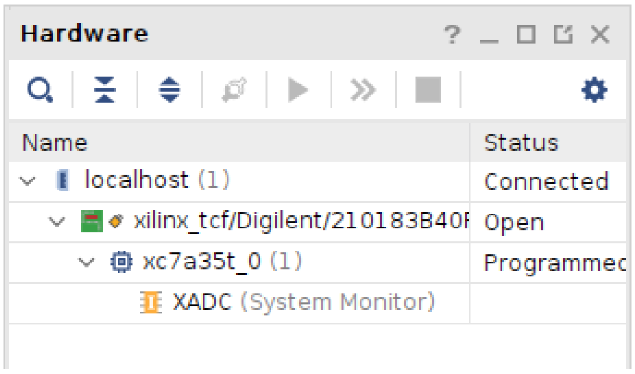
( ) Launch runs on Cluster lsf

( ) Generate scripts only

[ ] Don't show this dialog again

12. After this click on **Generate Bitstream**.

13. Once this is done, click on Open **Hardware Manager** => **Open Target** => **Auto Connect**. Ensure that basys3 board is connected to the system via micro-USB cable.

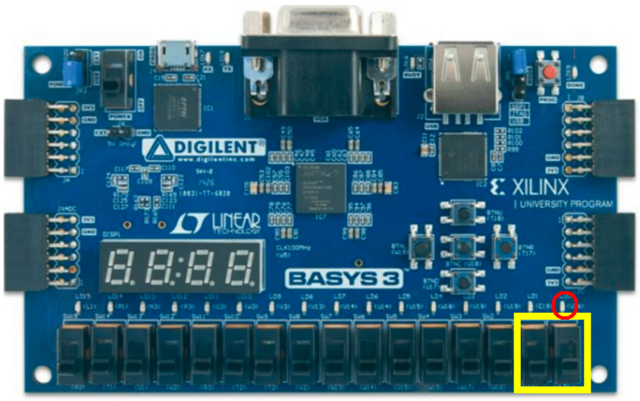


Hardware

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210183B40	Open
xc7a35t_0 (1)	Programme
XADC (System Monitor)	

14. Click on **Program Device**. Make sure the correct bit file is selected.

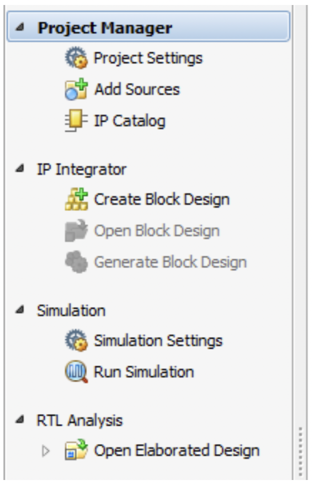
15. Once the process is complete, toggle the bottom right corner switches (V16 and V17, the AND gate inputs). This should change the LED (LD0/U16) according to the AND gate logic.

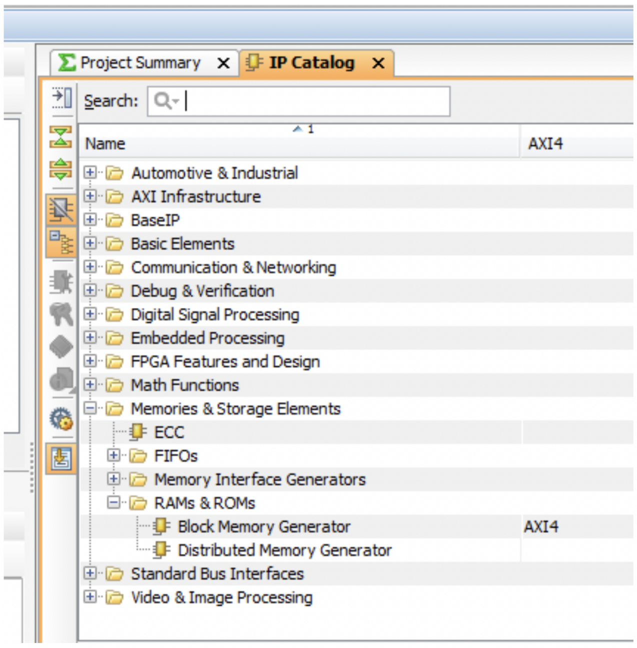


3 Memories

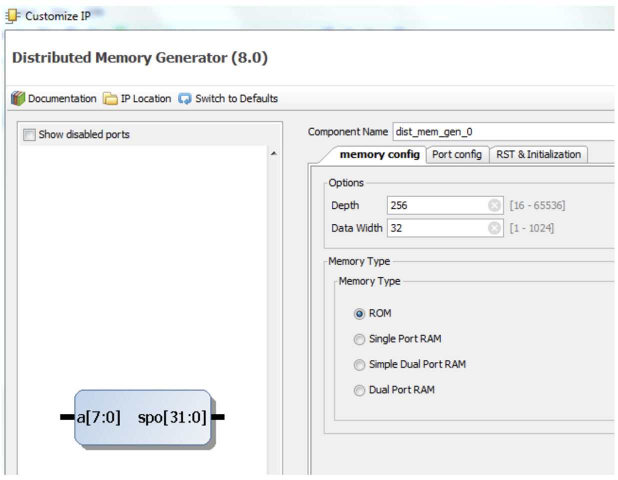
We would be using Vivado's memory generator to design program and data memories. Instructions are as follows:

- In Vivado, go to the Project Manager window on the left side of the screen which looks like the screen-shot shown on right.
- Open IP catalog. You can also open IP Catalog through a drop down menu after clicking on Window in the main menu on the top of the screen. "IP" (Intellectual Property) refers to pre-designed modules available for use. The IP catalog lists various IPs category-wise.
- Select the category **Memories & Storage Elements** and sub-category **RAMs & ROMs**, as shown below.





- Now choose **Distributed Memory Generator**, to create a ROM / RAM as per your design requirement. This generator creates memory modules of specified sizes using LUTs of FPGA. The reason for using the Distributed Memory Generator here is that **Block Memory Generator** creates a memory with an address register since they use FPGA BRAMs, and are mostly preferred when we want large on-chip memories.
  The distributed RAMs are faster and more efficient because they use the LUTs in contrast to FPGA BRAMs.
  The Distributed Memory Generator opens a window with three tabs. In the **memory config tab**, specify memory size and type. The screen-shot shown below is for a ROM with 256 words, each of size 32 bits.


Customize IP

Distributed Memory Generator (8.0)

Documentation | IP Location | Switch to Defaults

Show disabled ports

Component Name: dist_mem_gen_0

memory config | Port config | RST & Initialization

Options

Depth: 256 [16 - 65536]

Data Width: 32 [1 - 1024]

Memory Type

Memory Type

(●) ROM
( ) Single Port RAM
( ) Simple Dual Port RAM
( ) Dual Port RAM

- In the **RST & Initialization** tab, specify name of the file that defines the ROM contents. A sample file named *sample.coe* is available on moodle.
  Now instead of passing the inputs from the test bench or using "force value" construct, please initialize the inputs using a *coe* file and pass input to the DUT by connecting it to the memory module.
  Once you synthesize and implement your design, you will see a change in the number of LUTs used in your design.

4 Assignment Submission Instructions

Only one partner needs to submit. Mention all team member names and entry IDs during the submission.

1. Name the submission file as entryNumber1_entryNumber2.zip
2. Go to Gradescope
3. Upload the following files:
   - Source file (.vhd)
   - Constraint File (.xdc)
   - Bit file (.bit)
   - Report (.pdf) - The report needs to state your design decisions, lab work, simulation snapshots, and synthesis report (particularly resource counts: Flip-flops, LUTs, BRAMs, and DSPs)