Project 2
Due date: Fri, 22-Nov-2024 @ 11:59 PM
(No late submissions after 3PM on Mon, 25-Nov)

Learning Objectives:

- Learn to build RVfpga-EL2 from HDL with Vivado
- Learn to add peripherals to the RVfpga-EL2 Wishbone Bus
- Implement Video Animations
- Idea for Final project

1. Introduction

In this project you will add VGA peripheral to RVfpga-EL2 SoC that you build from the RVfpga source files using Vivado. The VGA Peripheral is customized for 480 x 640 @(31.5Mhz) resolution. You will implement two applications in C. You will use Catapult Studio to download the `.bit` file, create, and debug your applications.

This project has three parts:

- Develop the Wishbone interfaced VGA Peripheral in the RVfpga-EL2 sources
- Add number-to-pixel generation logic into the VGA Peripheral.
- Develop Application - 1 in C to replace even numbers from an array. Application -2: Design a video animation.

Project #2 can be done on either the RealDigital Boolean Board or the Digilent Nexys A7 (Nexys4 DDR). This project significantly involves learning about how the VGA-connected displays work and is an example of developing an embedded application using VGA. You will work in a team of 2. You may discuss this with other teams, but the code and report you submit must be your own.

2. Tasks

The tasks for this project make use of RVfpga-EL2 Labs 5 for generating the bitstream given for Project-1. The write-ups for the Boolean board were provided by the RVfpga EL2 development team.

We are supporting two FGPA development boards for ECE 540 this term:

- The Digilent NexysA7 (https://digilent.com/shop/nexys-a7-fpga-trainer-board-recommended-for-ece-curriculum/). This development board is the same as the Nexys4 DDR-100T. There are Nexys4 DDR boards in both the Capstone/Digital Systems Lab (FAB 60-04) and the **W**illow **C**reek **C**enter lab (WCC 313) in Hillsboro. The four workstations in the WCC Lab have Vivado 2022.2 and Catapult Studio installed. They are a viable development platform for you.

Important: Please logoff of WCC and Capstone Labs workstations when you leave. It requires a reboot of the PC to clear the user.

- The Real Digital Boolean (https://www.realdigital.org/hardware/boolean). The Boolean board

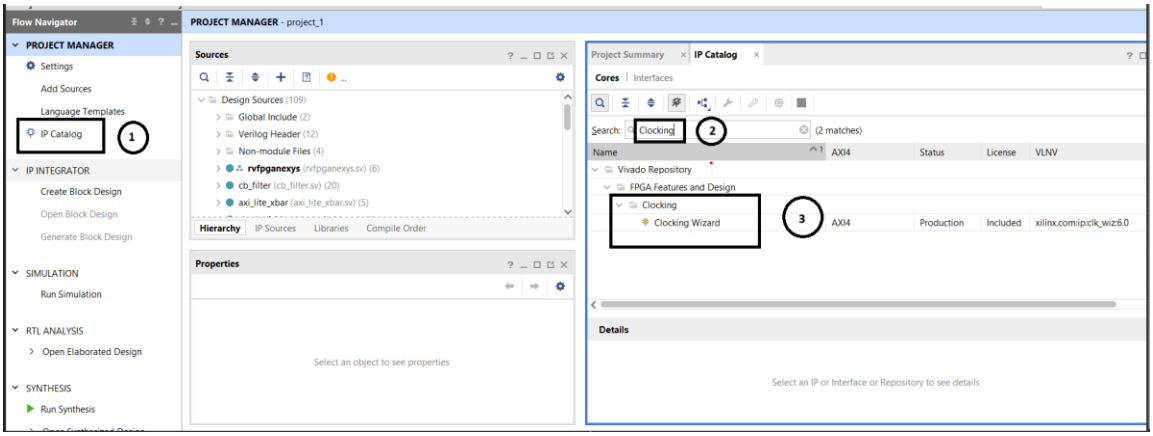
offers a much lower cost (and nicely featured) option for those planning to purchase their own development platform; it has sufficient FGPA resources for RVfpga and for a wide variety of SoC-based embedded application. I prototyped the ECE 544 Getting Started Project on my Boolean board and, as it turns out, the Microblaze MCS SoC was much smaller (and faster) than the RVfpga-based SoC we are using in ECE 540.

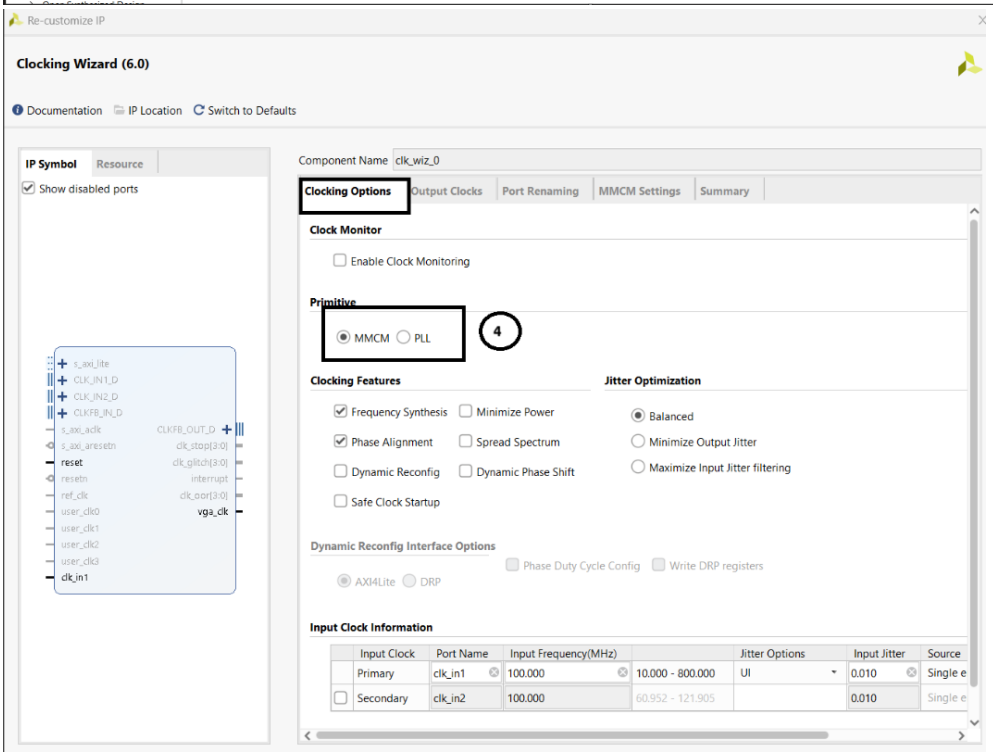
Note: *Please the new release (given for Project-1) of RVfpga EL2 for this project.*

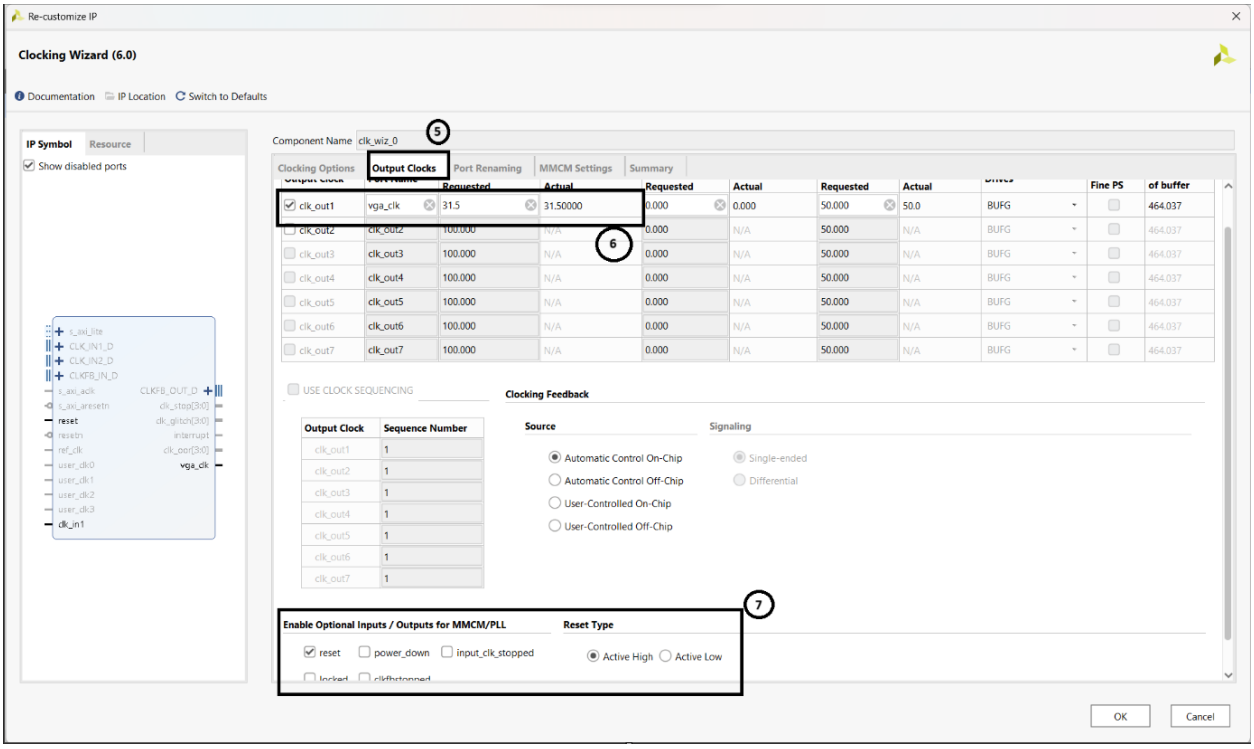
A. Part 1 – Add the VGA Peripheral into the EL2 Core

- Step - 1: Modify the system clock to 25MHz in swervolf core.
- Step - 2: Add a clock divider for the vga clk in rvfpganexys.sv.

Use the Clocking Wizard from the IP Catalog to create and customize the IP. Follow the steps in the images:







Click Ok and then generate the IP. After the IP is generated, go to Source window -> IP sources tab -> Instantiation template -> clk_wiz_0.veo and copy the module instantiation and use it in the rvfpganexys.sv. Modify the ports as follows:
input clk = clk; output clk = vga clk according to your design.

What is the operating frequency of the core?

- Step - 3: Create a folder in the Peripherals folder to create vga.v file (create the vga module in this file). Add the dtg.v file given into this folder. Add it to the Vivado Sources list using Add Sources option in Vivado.

- Step - 4: Create a new peripheral with address 0x80001500 in the wishbone and wire up all the wishbone signals till the top and vga module. Don't forget to make changes in your xdc file (Boolean users ignore this statement).
- Step - 5: (Boolean Users only) Copy the hdmi_tx_1_0 folder into the src folder. Add the modules present in hdl folder into the sources list in Vivado. Create a module instantiation in rvfpgaboolean.v and wire up the vga outputs to the module and HDMI outputs to the rvfpgaboolean module. Add the pinouts accordingly in the constraints.xdc file.

B. Part 2 – Number-to-Pixel generation logic

- Add the logic to read row, column and data as per the addresses below:

Addr[5:2]	Values

0x1	row = wb_data_i[19:10] column = wb_data_i[9:0]
0x3	Data_Buffer = wb_data_i[7:0]

- Develop the data to pixel conversion logic to display a character from the data buffer. **The data must be ascii characters of (0-9 and a-z)**. You are free to customise the size of the character to display (dont'make it too small).
  Ex: Data_Buffer = 0x1 (in ascii value) => pixel info would be ,

```
pixel_char = { 6'b000010,
               6'b001110,
               6'b000010,
               6'b000010,
               6'b111111}
```

C. Part 3 – Application - 1 and Application-2

- Application - 1: Write a C program to display characters from a given unsigned integer. The program needs to iterate over the array and display each number with equal width onto the display with a delay of about 1 sec(approx.). The condition here is that the even numbers should be displayed as 0 and the background has to be a red screen(intensity doesn't matter unless the characters are visible clearly).
- Application - 2: Write a C program to create any video animation or let's say a video wallpaper. Each group has to have a different animation.

  Both the programs should be in the same src location. To compile a particular file just change the CMAKE_LISTS.txt file and specify only the file and its location.

3. Deliverables

The deliverables for this project are enumerated in each of the parts. There are no deliverables for Part 1. Submit all the SystemVerilog/Verilog and constraint files you have modified for Part-2. For Part 3, submit your firmware repository.

Each source code file (SystemVerilog or application code) should start with a heading listing the author, the date created, and a short description of what the file contains and does. Clean up your code (e.g. no "dead" or commented-out code) before submitting. We will examine your source code and you'll leave a favorable impression if your code is readable and understandable. Neatness counts in ECE 540.

Please comment your code for Part-3. It helps us understand your logic.

Write/submit a 3–5-page design report for Part 2 and Part 3. Please describe all the problems that you have encountered during the Project and your steps to fix the issue. Add the snapshots of

the Application - 1 & 2 in your report.

Submit a short video showing each of your applications running. Narration is appreciated, but not required. You may submit one video for your entire project if you prefer. Pre-covid we conducted live demos which had both benefits and detriments. The pandemic caused us to rethink this and for the most part, students prefer to do a video than a live demo…and so do we. The final project is an exception – you will give a technical presentation on your project with a demo for the entire class.

All the Best for your Final Project!!