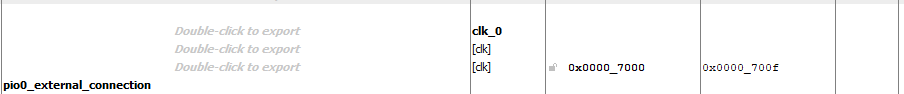
**LINGI 2315 - Homework 1 - My Linux App P1**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Delcoigne Ben | Noma | 38771700 |

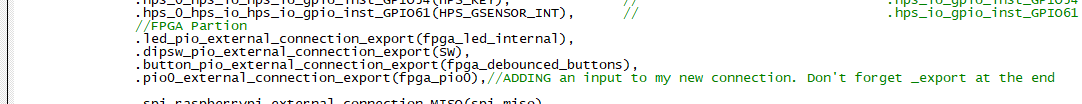
Description of the hardware modifications with relevant screenshots

In the platform designer I added a new input (26 bit PIO) which will later be used to set the led frequency. I mapped it to the first free address: 0x700

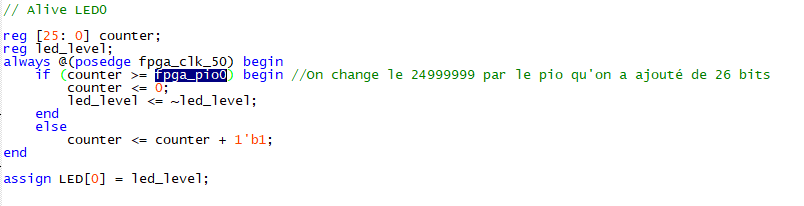
 

This was actually a part of the tutorial, I just set the length to 26 bits instead of 32.

In the HDL file, I added a line that “imports” that value, and I make it so that the clock reset time now depends on that value: 



Make the LED reset frequency depend on that value instead of just a hardcoded 2499999 value.

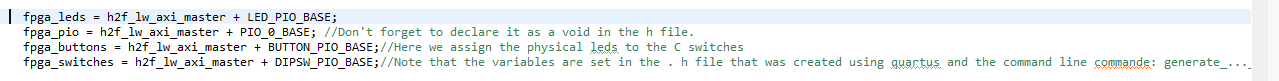


**LINGI 2315 - Homework 1 - My Linux App P2**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Delcoigne Ben | Noma | 38771700 |

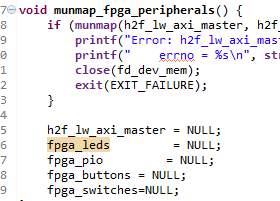
Description of the software modifications with relevant screenshots of your code

In the software, I first had to link the physical hardware IO’s to variables in the code. This is done with the following snippet: (part of the mmap\_fpga\_peripherals()) function.



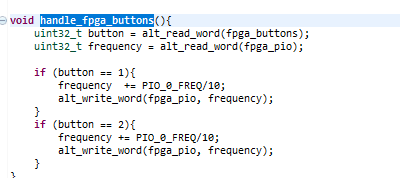
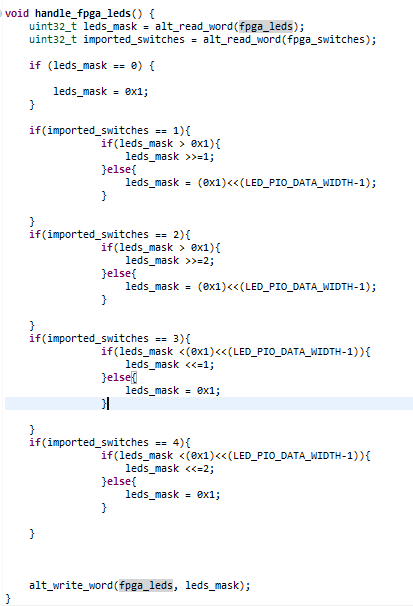
Note: I had to initialize the variables as void pointers in the .h file

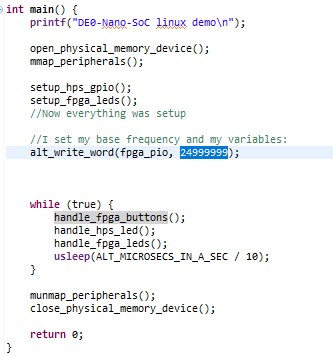
I also set these new variables to NULL in the unmap part of the code.



The classical LED code was changed to depend on the imported switches (used as a binary counter)

I also created a function that handles the buttons and changes the frequency of fpga\_pio\_0 (used in the Verilog code)





Finally, in the main code, I first hardcode a base frequency to fpga\_pio, and I also ask the code to activate the handle buttons function.

Here is a video demonstrating the workings of my homework:

<https://youtu.be/DXvES32BrSk>