NECSTlab Project - PYNQ API Generator

Pasquale Romano

Tutors: Emanuele Del Sozzo, Davide Conficconi

Contents

Introduction

1.1 Project aim

The aim of this project is to deliver a final Python script, able to generate a class, given an IP, that should help the final FPGA users to interact with it without necessarily knowing its implementation and hardware description or programming.

1.2 General specifications

The Python script can ask the user how he wants to insert the required data (different source code files' path) and then searches for the relevant addresses in these files through regexes. Regexes look for recurring patterns in strings. Source codes can be read as strings. In this way the script can read and extract the addresses, that can be later exploited to perform computations or just give the user an overview of the IP addresses setting.

Code insights

The script, after having acquired the proper files' paths from the user, firstly searches for the AXI interface that has been used (among Axi Lite, Axi Master and Axi Stream) as follows:

```
try:
    with open(cpp_file) as file:
        file_as_string = file.read().split("\n")
        for string in file_as_string:
            if top_function in string:
                i = file_as_string.index(string)
                file_as_string = file_as_string[i:]
        for line in file_as_string:
            if "pragma" in line:
                line = line.split(" ")
                for word in line:
                    if "axi" in word:
                        interface = axi_type[word]
                        break
```

Then it will search for and extract the main adresses:

```
try:
    with open(addresses_file) as file:
        file_as_string = file.read().split("\n")
        #reading addresses here. Not sure what to do with these
        for line in  file_as_string:
            if "//" not in line:
                break
            if "Control signals" in line:
                control_signals_offset = re.search('0x[0-9][0-9]', line)
            elif "Global Interrupt Enable Register" in line:
                global_interrupt_enable_register_offset = re.search('0x[0-9][0-9]', line)
            elif "IP Interrupt Enable Register (Read/Write)" in line:
                ip_interrupt_enable_offset = re.search('0x[0-9][0-9]', line)
            elif "IP Interrupt Status Register (Read/TOW)" in line:
                ip_interrupt_status_register_offset = re.search('0x[0-9][0-9]', line)
            elif "Data signal of" in line:
                letter = re.search('Data signal of ([a-z])', line)
                data_signals[letter[1]] = ''
except:
    print("Please enter a valid addresses file path\n")
    addresses_file = input()
```

So far, once the addresses are found, nothing is done with them, since some implementation space is left for future purposes.

Future implementation ideas

The script can be further implemented in order to provide other vary options to the final user. An idea could be to implement a script for classes of FPGAs that share common addresses settings (same number of variables, same types of variables...) and also to carry out functions that can wrap up existing PYNQ functions (store and read data, buffer creation, forming classes and so on).
Another option is to provide a GUI (Graphical User Interface) that can help create a Python class setting some parameters (like number and types of variables) and make the interaction easier in this way.