FINS-PY

An End of Summer Recap

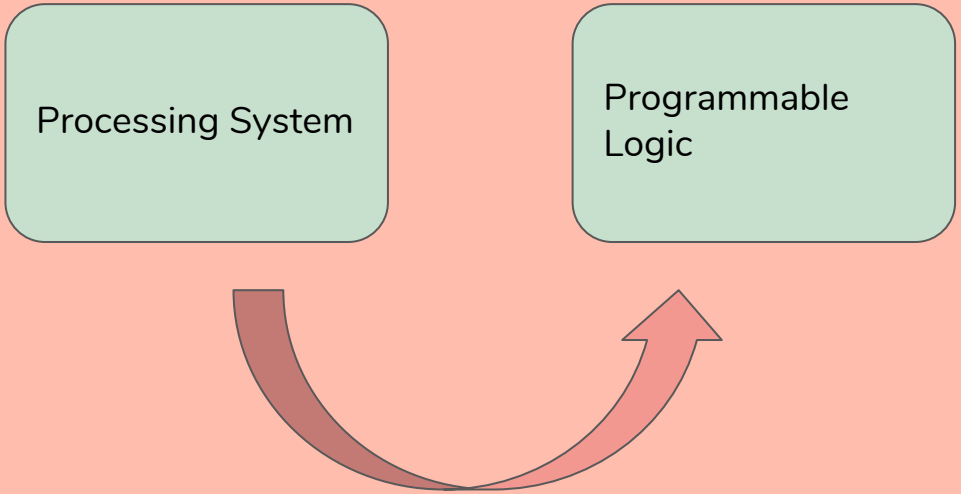
Michelle

Table of Contents

01

Project Overview

Background



SoC (System on a chip) architecture

FINS
(fins-sw and fins-sw-devmem)

The libraries

FINS

fins-sw

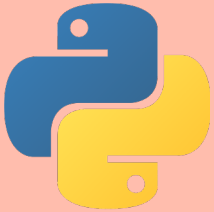
-a set of base classes with virtual functions
-Describe the properties/registers in the Programmable Logic
-C++

fins-sw-devmem

-Interacts with /dev/mem to gain access to physical address space
-C++

The Project

- Develop python package 'fins-py' as "runtime" interface
- Execute the same operations as fins-sw and fins-sw-devmem
- Will not require recompiling



02

Project Status

Progress

- fins-py package created and uploaded to Curiosity
- Most functions from C++ implemented
- fins-py.py
- pydevmem.py

fins-py.py:

FinsNodeset → FinsNode → FinsProperty

Class FinsNodeset

- Loads the JSON file and calls on the FinsNode class for each node in the file
- Holds each FinsNode in a list

- get_property_description()
- get_property_length()
- list_nodes()
- list_properties_of_node()
- dump_properties_of_node()
- test_default_values()
- test_write_width()
- load()
- check_loaded()
- print_get_node()
- create_nodes()
- add_list_item()
- get_node()
- length()
- output_list()
- get_node_by_name()
- read()
- write()
- get_property_from_node_name_or_id()

Class FinsNodeset

```
def load(self, filepath):
    #loads the file
    my_file = open(filepath, "r")
    try:
        data = json.load(my_file)
    except ValueError as err:
        raise ValueError("The file must be JSON")

    global loaded
    loaded = True
    my_file.close()
    self.base_offset = data['base_offset']
    self.name = data['name']
    node1 = data['nodes']                          #th
    self.create_nodes(node1)
```

```
def create_nodes(self, input):
    i = 0
    for i in range(len(input)):
        fin_node = FinsNode()
        fin_node.set_properties(input[i])
        self.add_list_item(fin_node)
```

FinsNode() is called for each node from the JSON

Class FinsNode

- Holds the field attributes of a node from the JSON
- Calls on the FinsProperty class for each array of properties

- print_node_info()
- set_properties()
- gets_fins_property_by_name()
- property_length()

```
def set_properties(self, file): #pass in the node and set the node properties
    #file = 1 node (dictionary)

    self.node_id                = file['node_id']
    self.node_name              = file['node_name']
    self.ports_consumer         = file['ports_consumer']
    self.ports_consumer_name    = file['ports_consumer_name']
    self.ports_producer         = file['ports_producer']
    self.ports_producer_name    = file['ports_producer_name']
    self.properties_offset      = file['properties_offset']
    self.module_name            = file['module_name']
    self.fins_path              = file['fins_path']
    self.interface_name         = file['interface_name']

    plist = file['properties'] #get the fins_property of values from the node
    #for all the sets of properties in the node
    for i in range(0, len(plist)):

        finp = FinsProperty(plist[i])

        self.fins_property.append(finp)
```

Class FinsProperty

- Holds the field attributes of each property array from a node in the JSON file
- read() calls on pydevmem

```
class FinsProperty:
    def __init__(self, prop):

        try:
            self.default_values = prop['default_values']
            self.name           = prop['name']
            self.description    = prop['description']
            self.width          = prop['width']
            self.is_signed      = prop['is_signed']
            self.length         = prop['length']
            self.range_min      = prop['range_min']
            self.range_max      = prop['range_max']
            self.offset         = prop['offset']
            self.is_writable    = prop['is_writable']
            self.is_readable    = prop['is_readable']
            self.type           = prop['type']
        except:
            print("An error occurred!")
```

- list_all_properties()
- get_default_values()
- read()
- write()
- get_property()
- get_name()
- get_description()
- get_width()
- get_is_signed()
- get_length()
- get_range_min()
- get_range_max()
- get_type()
- get_offset()
- get_is_writable()
- get_is_readable()

Progress

- Moved onto testing phase with zedboard
- Initial problems with set up
- Overall: 70% project completion

```
@dispatch(str)
def read(self, property_name): #given only the FP's name
    current_node = self.head
    current_fins_property = None
    while current_node is not None:
        for i in range(0, current_node.property_length()):
            current_fins_property = current_node.fins_property[i]
            current_name = current_fins_property.get_name()
            if current_name == property_name:
                return current_fins_property.read(current_node.properties_offset, self.base_offset)
        current_node = current_node.next
```

03

Challenges

Challenges

- Understanding the files in FINS
- Translating from C++ to Python
- Working from home
- Issues with hardware set up

04

What I learned

What I learned

- Revisiting Python
- Revisiting C++
- /dev/mem and architecture
- Keep documentation
- Trial and Error
- Remote work

Thank you!