Python 在射频测试设备自动化中的应用

阶段汇报

20300750008 谢承臻

国家集成电路创新中心

2023 年 11 月 21 日

復旦大學
Fudan University

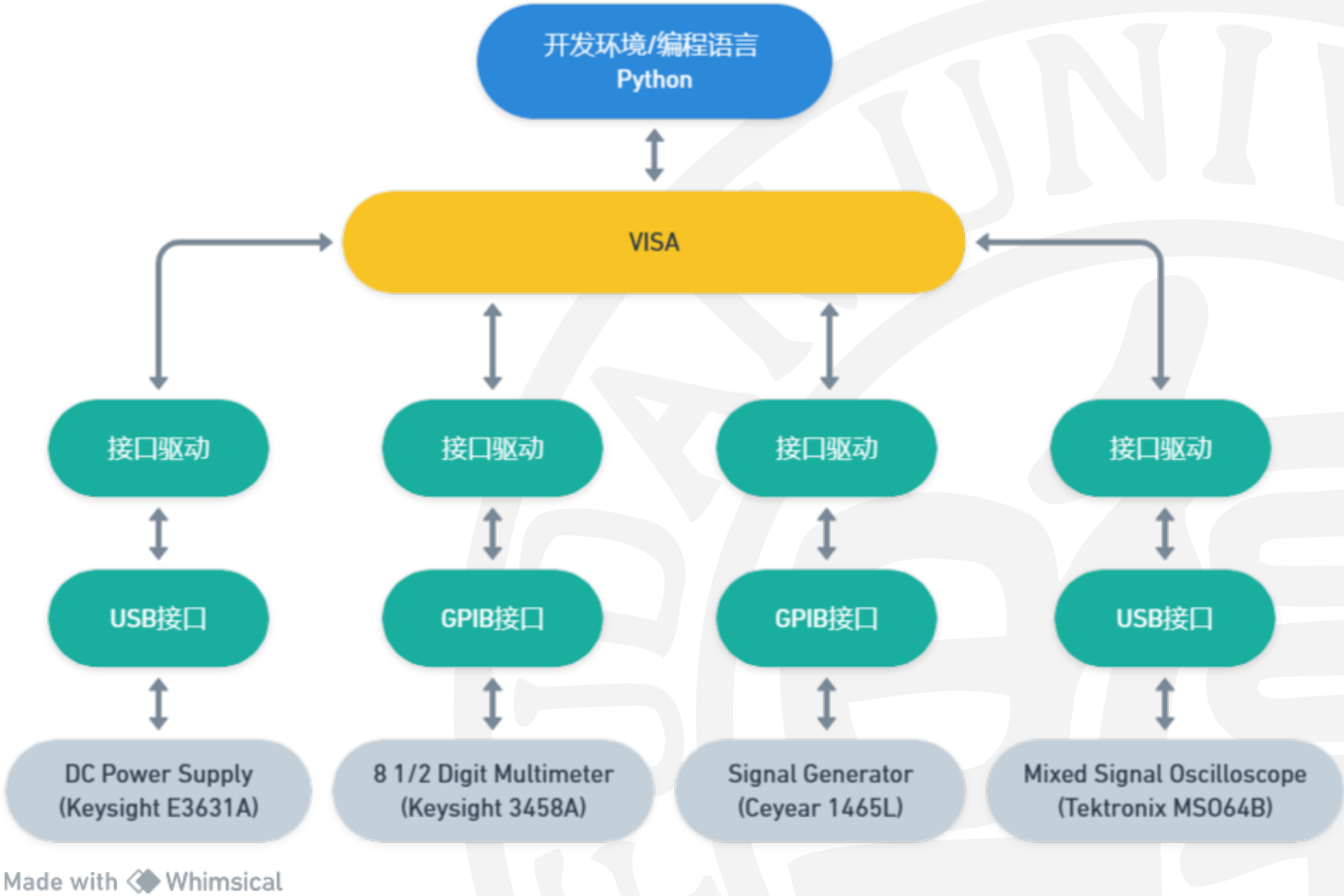
研究背景

- 提高自动化测试效率
  - 能够更快速、一致地执行测试案例，节省人工成本
  - 减少人为错误，提高测试准确性和可靠性
- 提高测试覆盖率
  - 更容易覆盖大量的测试用例，包括一些难以手动测试的情况，有助于确保测试的准确性和可靠性
- 适应多样化的测试需求

Why Python

	LabView	Matlab	C++	tcl	python
免费	N	N	Y	Y	Y
交互式	N	Y	N	Y	Y
方便性	N	Y	N	Y	Y
拓展性	Y	Y	Y	N	Y

程控软硬件层


开发环境/编程语言
Python
VISA
接口驱动
接口驱动
接口驱动
接口驱动
USB接口
GPIB接口
GPIB接口
USB接口
DC Power Supply
(Keysight E3631A)
8 1/2 Digit Multimeter
(Keysight 3458A)
Signal Generator
(Ceyear 1465L)
Mixed Signal Oscilloscope
(Tektronix MSO64B)
Made with Whimsical

VISA

Virtual Instrument Software Architecture

VISA[1]是标准的 I/O 函数库及其相关行业规范的总称，由 Keysight Technologies, National Instrument, Rohde & Schwarz, Tektronix 等多家公司确定，广泛用于仪器与电脑通信与自动化测试方案。

VISA 库函数是一套可方便调用的函数，其核心函数能够控制各种类型器件，无需考虑器件的接口类型[2]和不同 I/O 接口软件的使用方法。这些库函数用于编写仪器的驱动程序，完成计算机与仪器见的命令和输出传输，以实现对仪器的程控。

[1] see some more through *link*

[2] 支持 VXI、GPIB、LAN 及 USB 等接口

项目结构

```
RF_automation
├── .git
├── README.md
├── TEST
│   ├── data
│   ├── example.py
│   └── list_resources.py
├── modules
│   ├── DCPowerSupply_ES3631A
│   │   ├── DCPowerSupply_ES3631A.py
│   │   ├── __init__.py
│   │   └── __pycache__
│   ├── Multimeter_3458A
│   │   ├── Multimeter_3458A.py
│   │   ├── __init__.py
│   │   └── __pycache__
│   ├── Oscilloscope_MSO64B
│   │   ├── Oscilloscope_MSO64B.py
│   │   ├── __init__.py
│   │   └── __pycache__
│   ├── SignalGenerator_1465L
│   │   ├── SignalGenerator_1465L.py
│   │   ├── __init__.py
│   │   └── __pycache__
└── references
```

- 各测试仪器代码都已模块化，放在./module 下供调用
- 测试代码和结果数据等放在./TEST 下

SCPI

Standard Commands for Programmable Instruments

- SCPI 于 1990 与 IEEE 488.2 协议一起面世，这套标准定义了可用于控制一切仪器的语法，命令结构以及数据格式
- 不同设备有共同的指令，比如 `*IDN?`, `*RST`, `*OPC?` 等
- 大部分特异化的指令在设备的编程指南中可以找到使用方法
- 有些在 1990 前出现的设备可以试试找现成的 python 库

设备连接

```
1 import pyvisa as visa
2 rm = visa.ResourceManager()
3 print(rm.list_resources()) # 显示连接上了的设备地址
4 resource_name = 'GPIB0::22::INSTR'  # 设备连接地址
5 scope = rm.open_resource(resource_name)
6 scope.write('*RST')              # 在scope输入指令
```

例：数字万用表读取仪表盘示数

```
1  class Multimeter_3458A:
2      def __init__(self, resource_name):
3          self.rm = visa.ResourceManager()
4          self.scope = self.rm.open_resource(resource_name)
5          self.scope.write('END␣ON')
6
7          ID_msg = self.scope.query_ascii_values('ID?', converter='s', separator='\r\n')
8          print("\n␣␣Connected␣to␣Multimeter:␣", ID_msg[0])
9
10     def get_data(self):
11         time.sleep(DEFAULT_TIMEOUT)
12         list = self.scope.query_ascii_values('READ?', converter='s', separator='\r\n')
13         data = list[0]
14         data = float(data.replace("␣", ""))
15         return data
16
17     def close(self):
18         self.scope.close()
19         self.rm.close()
```

例：数字万用表读取仪表盘示数

```
1 MM_resource_name = "GPIB0::22::INSTR"
2 MM = Multimeter_3458A(MM_resource_name)
3 # ..... 中间设置好了DCV和量程
4 Volt_output = MM.get_data()
5 print("␣␣␣␣Voltage␣output␣=␣", Volt_output, "V")
6 MM.close()
```

测试案例——test LDO

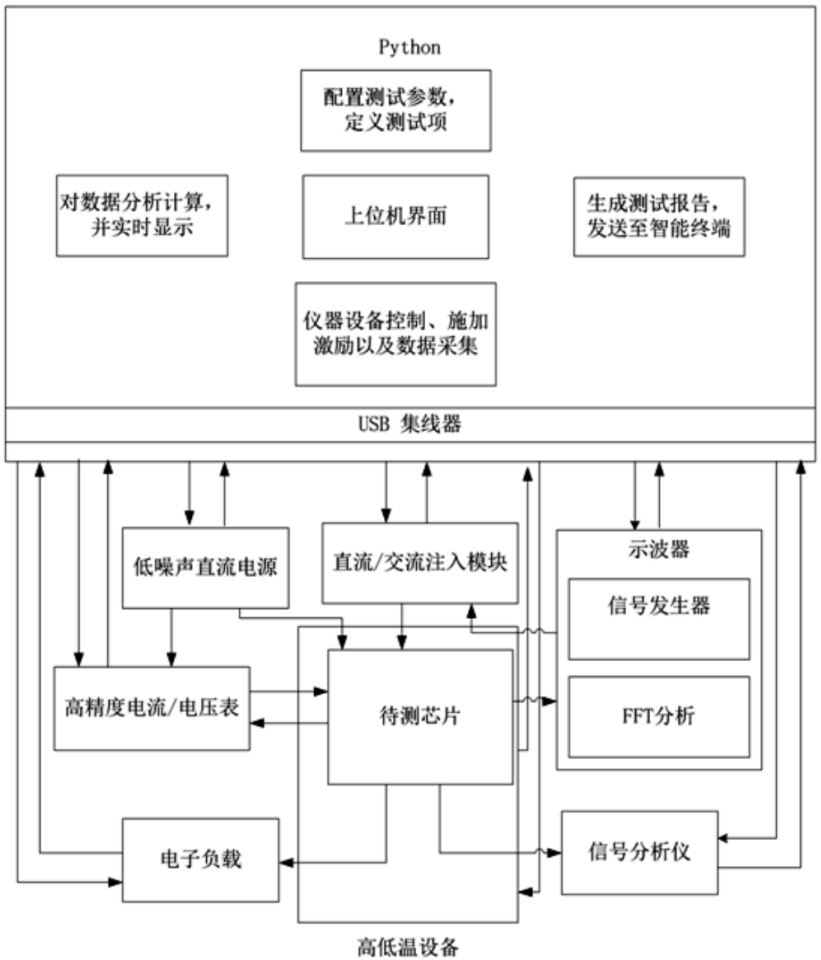


图 1: LDO 智能测试系统框图

测试案例——test LDO

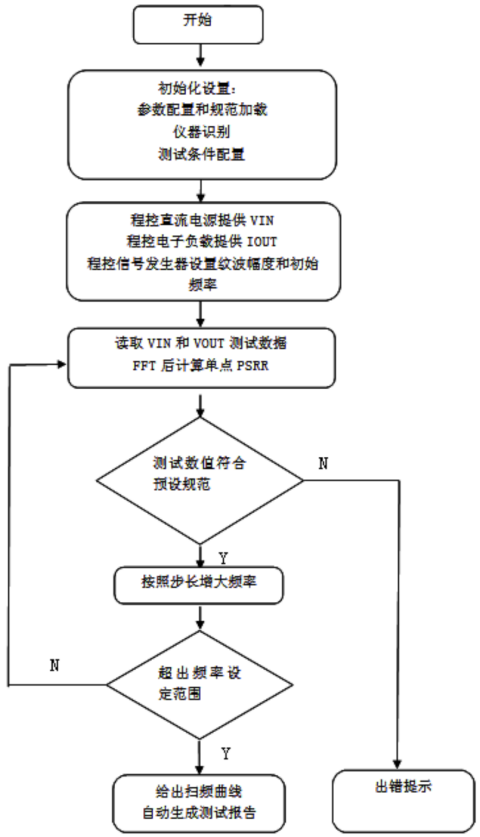


图 2: PSRR 测试流程图

Thanks!