The Journey of Google Taiwan

Yi-Yang Lin

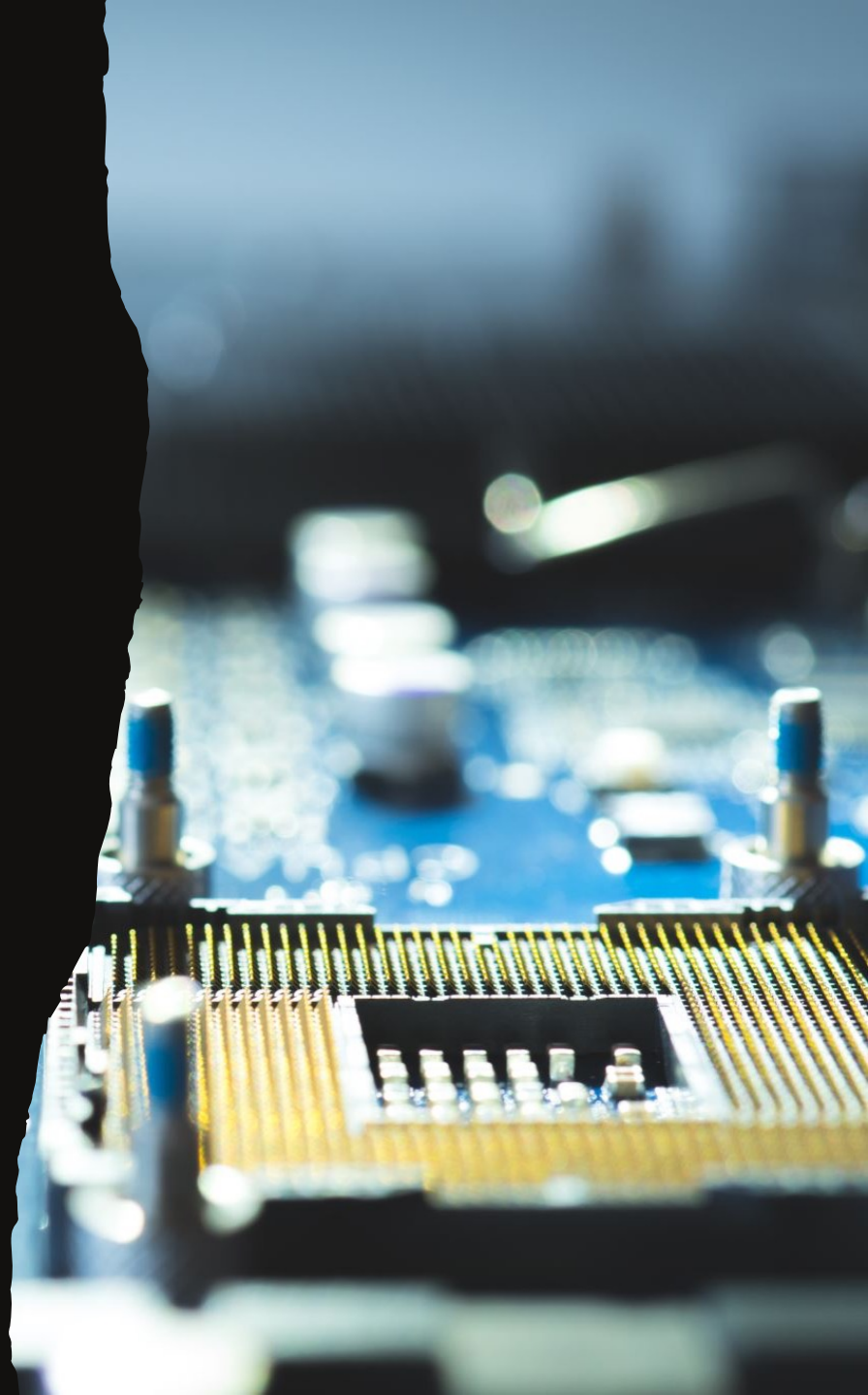
About me

Education

- Boston University
  - Master of Science, Computer Science
- University of Wisconsin-Madison
  - Bachelor of Science, Computer Science
- Bellevue College
  - Associate of Arts and Sciences with Distinction, Computer Science

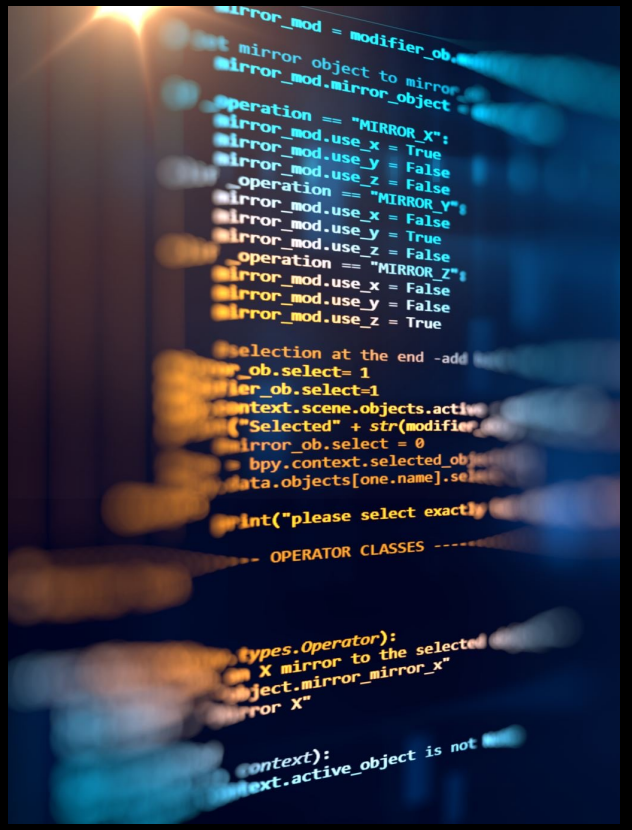
Software Engineer at Intel Taipei, 2021

- Cross-validation-testing Team in Wireless Connectivity Solution
- Implemented and Maintained Software Regression Automation Validation Tools
- Maintained automation Cloud-based Infrastructure Titan to monitor each RF shielded station.
    - Created automatic scripts and wrote documentations to resolve configuration issues.
    - Set up testing operating system by scripting languages and arranged RF shielded room environments.
- Intel Wireless Adapter Driver Validation
    - Delivered accurate develop team.
    - Differentiated issues affected by which develop teams and discussed on JIRA platform.
- Tested OSI Physical Layer of Wireless Adapter Driver (Bluetooth/WiFi) with Microwave Analyzer in WCS Lab
    - Ensured daily regression delivered expected performance with each dev commit.
    - Clarified physical issues not related daily regression results from 100+ PC stations located in Israel, India, Taiwan to code to driver performance.
    - Researched the Transmission and Reception performance with testing flows in HIT Manager and Keysight equipment.



Engineering Intern at Google Taipei, Summer 2023

- RF Software Team in Pixel Modem Wireless
- Designed Python Script for merging C++ Configuration files, inputting to signaling equipment.
- Implemented infrastructure tools *Piper*, *google3*, *Critique* for designing Python Script, connecting input data for Pixel RF testing equipment.
- Utilized SQL querying functions from automation framework imported in *Spanner* database to generate combo key data structures.
- Created 2 Technical design docs for code reference
- Presented 5G communication topic *Registration Procedure* with recorded videos.



Social Links

- https://www.linkedin.com/in/yiyang-lin/
- https://github.com/yiyanglin0102/
- https://www.facebook.com/yiyanglin0102/
- https://yiyanglin.com/

Google Work

Pixel Mobile Wireless
L1 RF Software

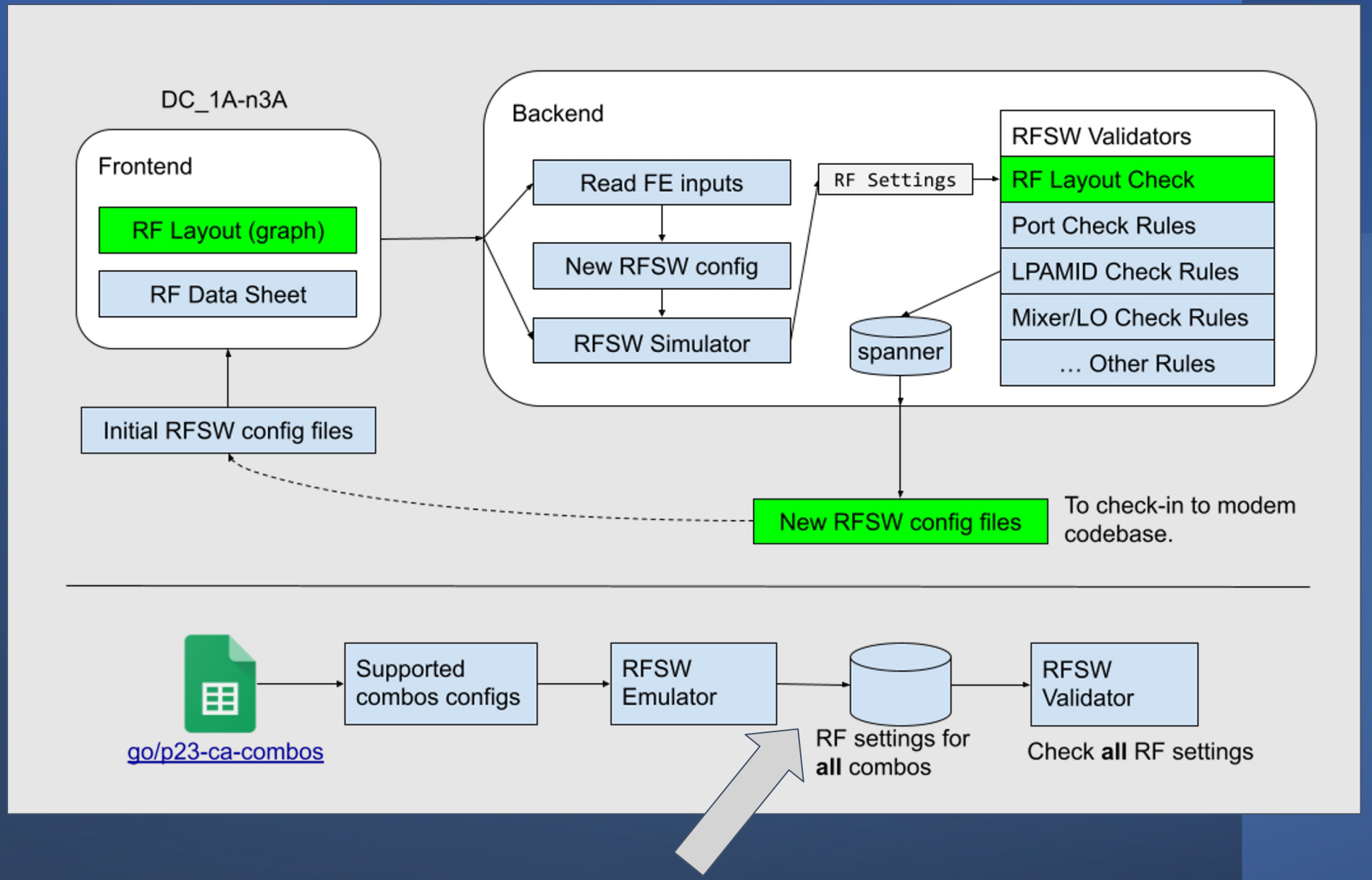
Intern Project

POR Testcase Script Generate

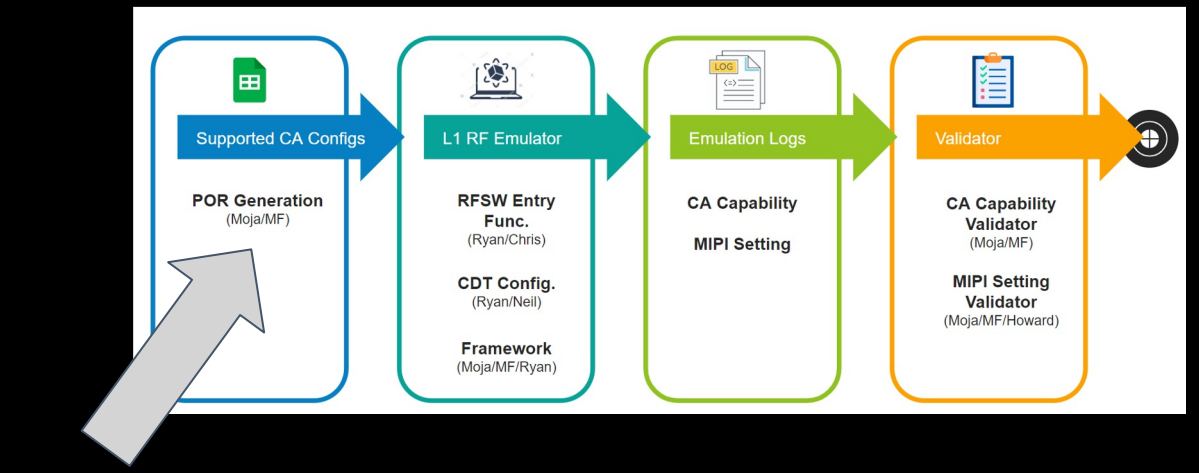
Outline

- Background
- Objective
- Source Code
- Code Build Process
- Testcase Generation from POR for L1 RF Regression Code Review
    - Trace Code
- RF Terminology

Background


DC_1A-n3A
Frontend
RF Layout (graph)
RF Data Sheet
Backend
Read FE inputs
New RFSW config
RFSW Simulator
RF Settings
RFSW Validators
RF Layout Check
Port Check Rules
LPAMID Check Rules
Mixer/LO Check Rules
… Other Rules
spanner
Initial RFSW config files
New RFSW config files
To check-in to modem codebase.
go/p23-ca-combos
Supported combos configs
RFSW Emulator
RF settings for all combos
RFSW Validator
Check all RF settings

Testcase Generation from POR for L1 RF Regression

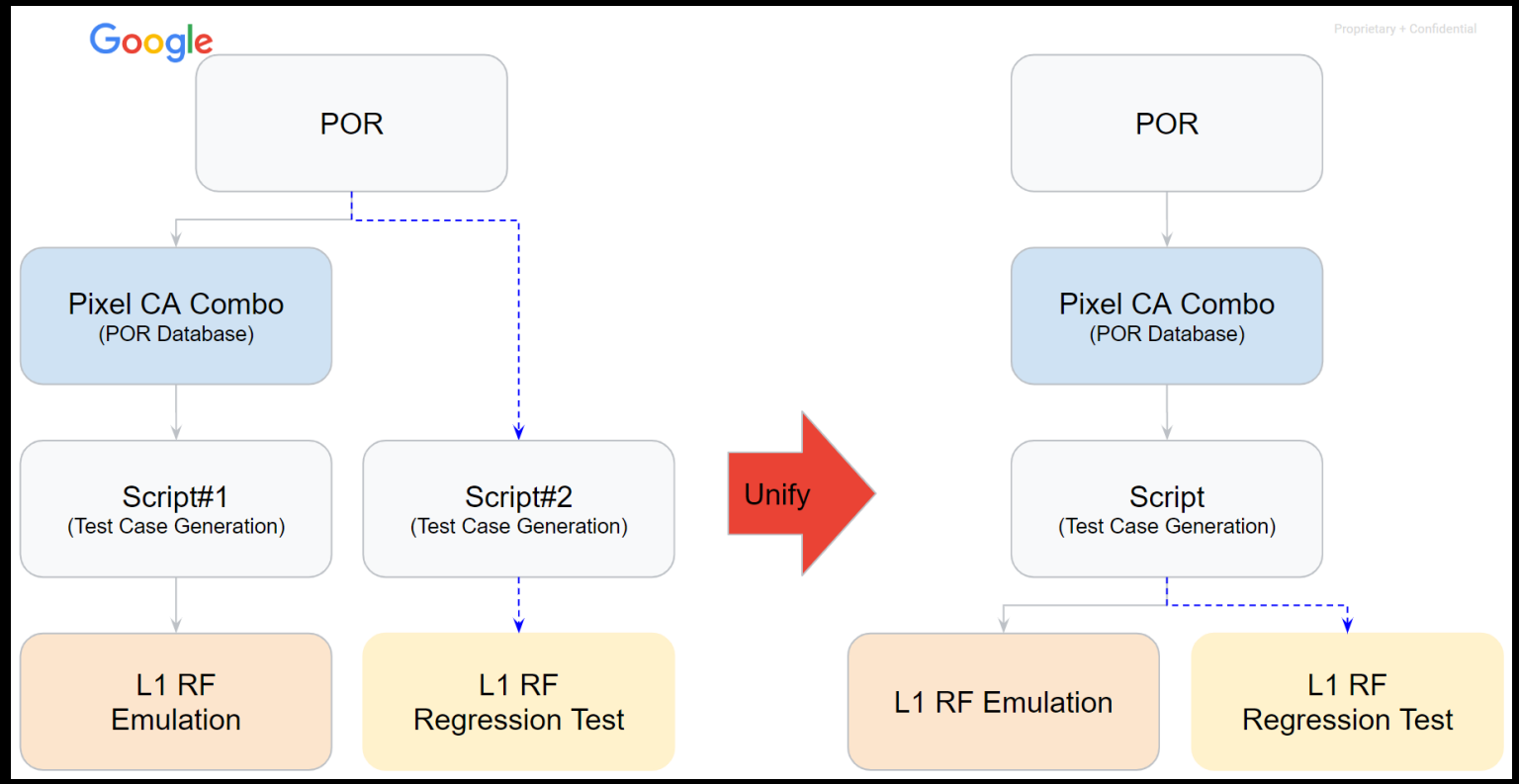


Emulator input from POR

Objective

- **Creating script for connecting POR to**
- **L1 RF Regression Tests**
- **By adding methods in rf_ca_config_generator_main.py**
- ***Main***
- **Merging L1 RF Emulation and L1 RF Regression Test Scripts**

L1 RF Regression Input File Format



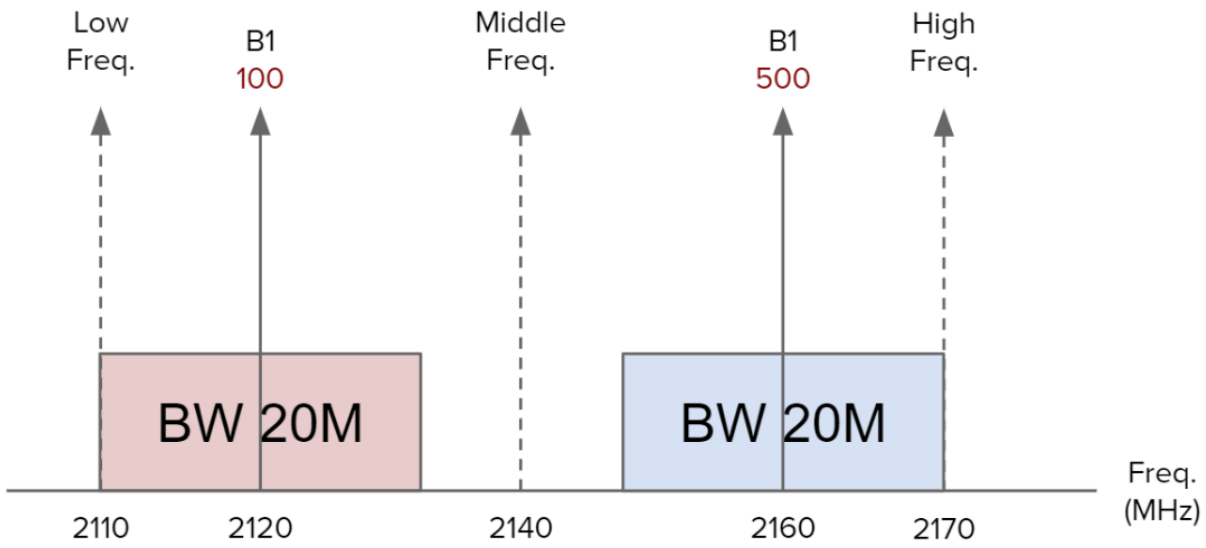
POR

go/l10-ca-combos

To see what is in the codebase (Should match!) see go/ca-combos-dash	US mmW SKU
Total=13623	
Configuration Name	**Default UL and MIMO config US mmW SKU**
DC_1A-1A-28A_n7A	DC_28A_n7A(2-2-2-2)[[20]]

Test Case

28A-1A-1A_n7A#(28A-1A-1A,9310-100-500,20M-20M-20M,2-2-2);(n7A,531000,50M,2)



Format :

Combo_Name #

(

LTE_Band#1-.....-LTE_Band#n ,

EArfcn#1-.....EArfcn#n ,

BW#1-.....-BW#n ,

MIMO#1-.....-MIMO#n

)

;

(

NR_Band#1-.....-NR_Band#n ,

NR-Arfcn#1-.....NR-Arfcn#n ,

BW#1-.....-BW#n ,

MIMO#1- .....-MIMO#n

)

```
combo_info_db = (
    ComboInfo("1A", [[300]], [[20]]),
    ComboInfo("1A-n1A", [[100], [430000]], [[20], [20]]),
        .
        .
        .
        .
        .
    ComboInfo("n78C", [[633334, 639998]], [[100, 100]]),
    ComboInfo("n78A-n78A", [[650000], [623334]], [[100], [100]]),
    ComboInfo("n79A", [[713334]], [[100]]),
)
```

Generated L1 RF Regression Input File Format

- **8A - 3A_n41A # (8A-3A, 36525-1575, 10M-20M, 2-4);(n41A, 518598, 100M, 4)**

Format :

Combo_Name #

(

LTE_Band#1-.....-LTE_Band#n ,
EArfcn#1-.....EArfcn#n ,
BW#1-……-BW#n ,
MIMO#1-……-MIMO#n

)

;

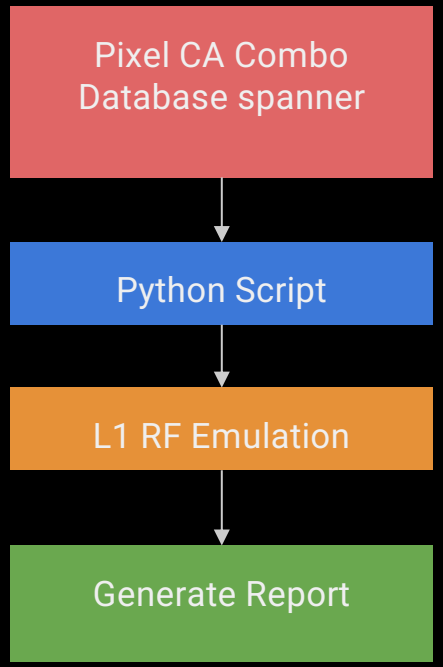
(

NR_Band#1-.....-NR_Band#n ,
NR-Arfcn#1-.....NR-Arfcn#n ,
BW#1-……-BW#n ,
MIMO#1- ……-MIMO#n

)

Testcase Generation from POR for L1 RF Regression

Steps prior to generating report

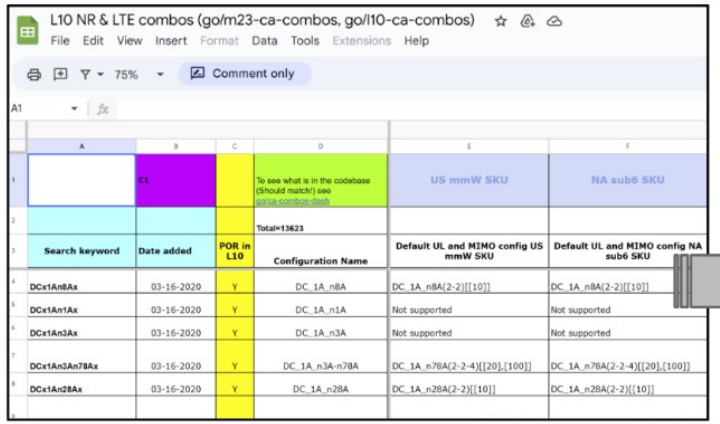


ca_configs.cpp

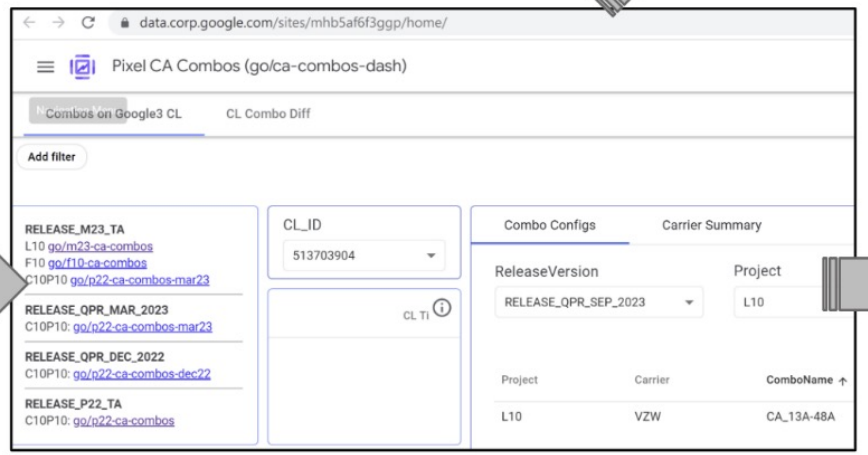
```
extern "C" {
#include "ca_configs.h"
}

const CaConfig_t ca_12_25_p12 = {
    .name = "ca_12_25_p12",
    .band_cnt = 2,
    .bandConfigList = {
        {RF_BAND_12, false, true},
        {RF_BAND_25, false, false},
    },
};
```

POR



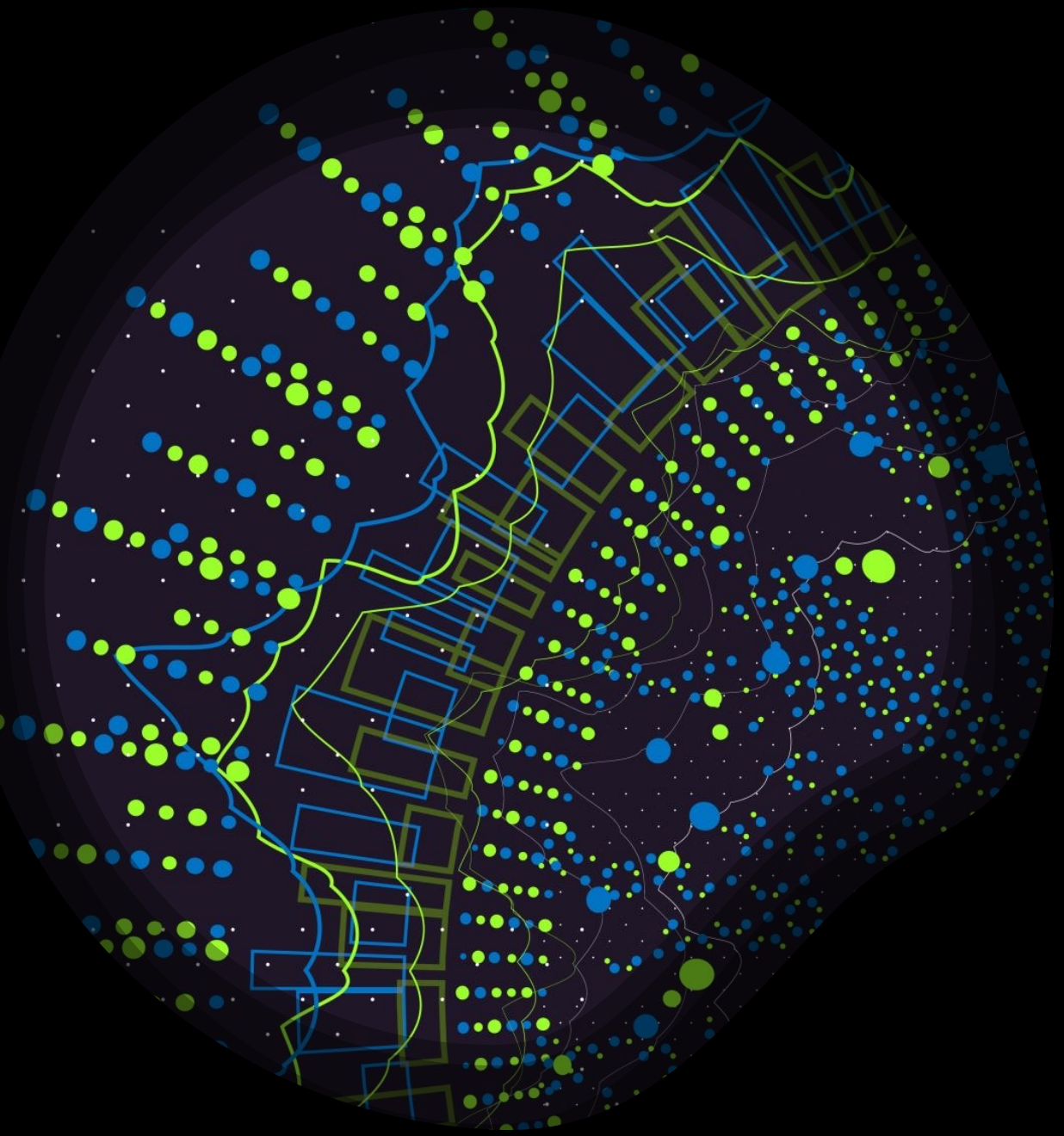
Pixel CA Combos



l10_mmw.cpp

```
#include <vector>
extern "C" {
#include "ca_configs.h"
}

std::vector<const CaConfig_t *> l10_sub6_row_ca_configs {
    &ca_12_25_p12,
    &ca_12_2_4_p12,
    &ca_12_2_66_p12,
    &ca_12_2_7_p12,
    &ca_12_2_p12,
    &ca_12_4_7_p12,
    &ca_12_4_p12,
    &ca_12_66_p12,
    &ca_12_7_66_p12,
    &ca_12_7f_p12,
    &ca_12_p12,
    &ca_17_2_p17,
```

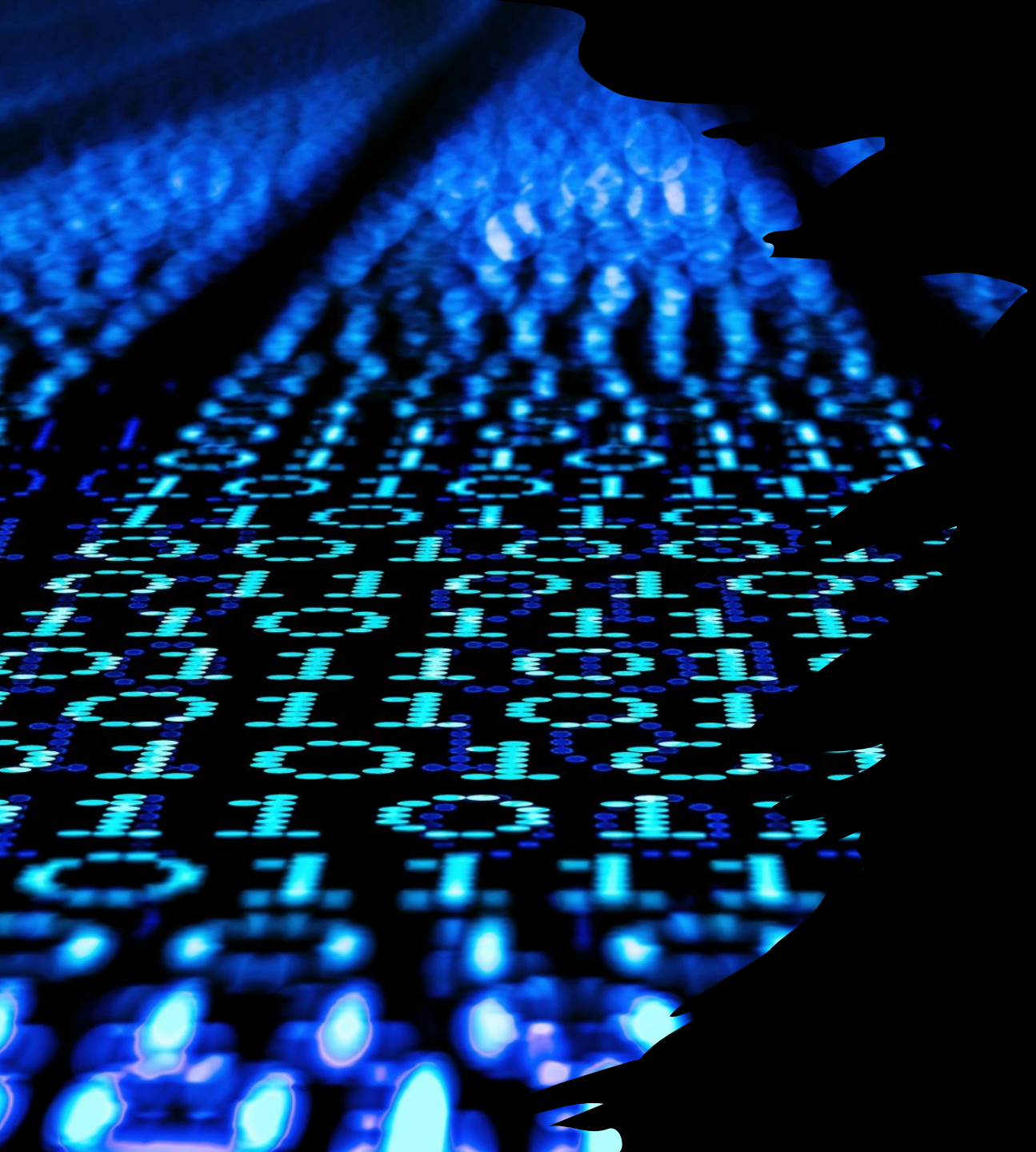


Infrastructure and Tools

- **Editor**
  - **Cider-V**
- ***Lab go/dwf***
- **Get access**
  - **gcert ssh**
  - **Corp Normal**
- **Cloudtop**
  - **Chrome Remote Desktop**
- **Database**
  - **Spanner**
- **Blaze**
- **google3**
  - ***Piper***
  - ***/google3/wireless/android/pixel/modem***
  - ***/ca_combo_gen/ca_combo_gen/rf_ca_config_gen/***

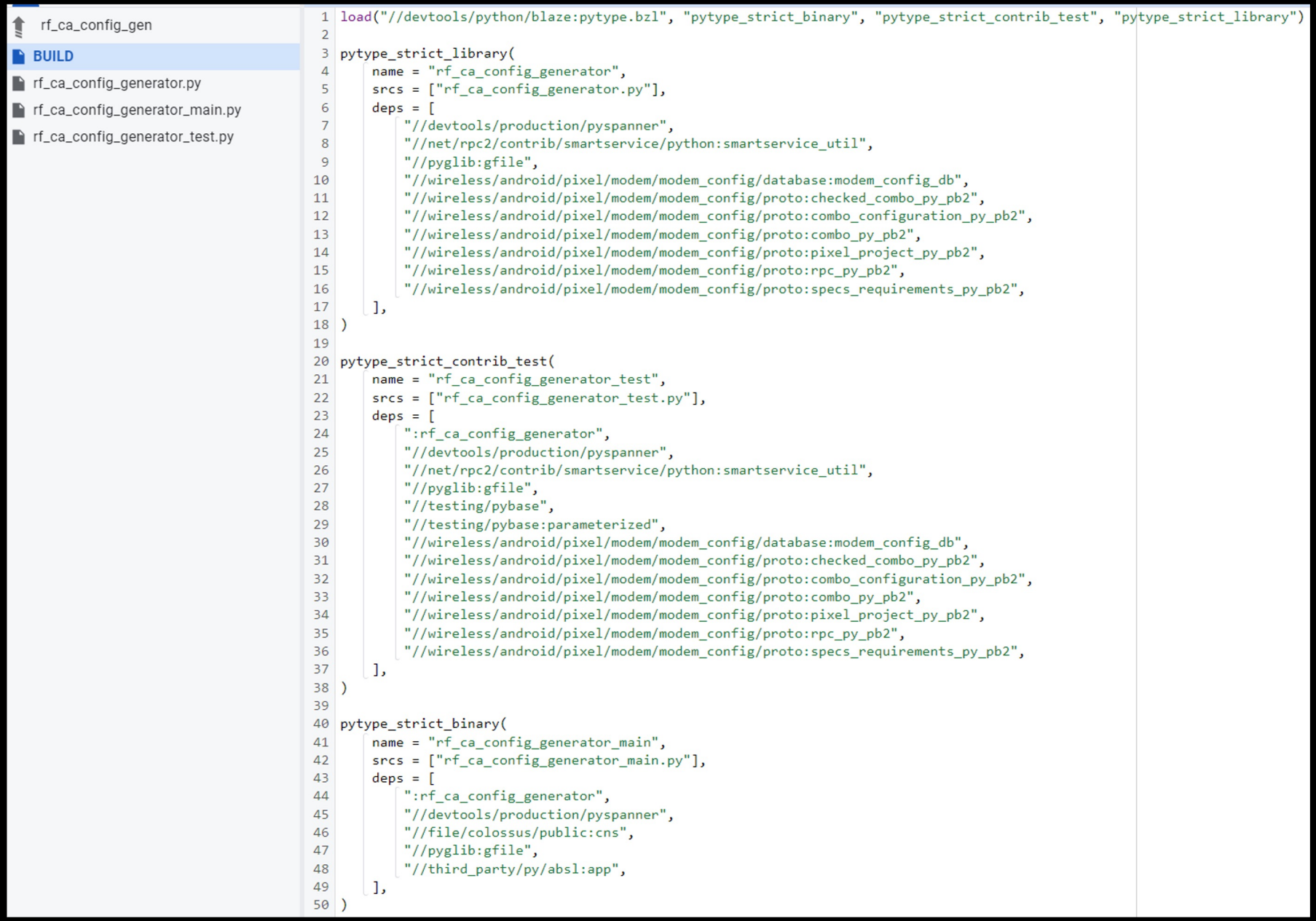
Build Process

- **Understand Loading Dependencies from google3**
  - **proto/ .proto files**
  - **Database - Spanner**
- **Blaze**
  - **Command under the directory**
  - ***/google/src/cloud/username/workspace/google3/wireless/android/pixel/modem***
  - ***/ca_combo_gen/ca_combo_gen/rf_ca_config_gen***
    - ***blaze run rf_ca_config_generator_main***
- **BUILD File Structure**



BUILD File

- Containing the rules that *Blaze* uses to build that directory and its children into the package.
- A list of targets, their sources, and their dependencies describing rules.
- Similar to *Makefile* in Unix / Linux
- Import dependencies from Infrastructure *proto*, *spanner* … etc.
- Main entry point for Python build configuration workflow


rf_ca_config_gen
BUILD
rf_ca_config_generator.py
rf_ca_config_generator_main.py
rf_ca_config_generator_test.py
load("//devtools/python/blaze:pytype.bzl", "pytype_strict_binary", "pytype_strict_contrib_test", "pytype_strict_library")
pytype_strict_library(
name = "rf_ca_config_generator",
srcs = ["rf_ca_config_generator.py"],
deps = [
"//devtools/production/pyspanner",
"//net/rpc2/contrib/smartservice/python:smartservice_util",
"//pyglib:gfile",
"//wireless/android/pixel/modem/modem_config/database:modem_config_db",
"//wireless/android/pixel/modem/modem_config/proto:checked_combo_py_pb2",
"//wireless/android/pixel/modem/modem_config/proto:combo_configuration_py_pb2",
"//wireless/android/pixel/modem/modem_config/proto:combo_py_pb2",
"//wireless/android/pixel/modem/modem_config/proto:pixel_project_py_pb2",
"//wireless/android/pixel/modem/modem_config/proto:rpc_py_pb2",
"//wireless/android/pixel/modem/modem_config/proto:specs_requirements_py_pb2",
],
)
pytype_strict_contrib_test(
name = "rf_ca_config_generator_test",
srcs = ["rf_ca_config_generator_test.py"],
deps = [
":rf_ca_config_generator",
"//devtools/production/pyspanner",
"//net/rpc2/contrib/smartservice/python:smartservice_util",
"//pyglib:gfile",
"//testing/pybase",
"//testing/pybase:parameterized",
"//wireless/android/pixel/modem/modem_config/database:modem_config_db",
"//wireless/android/pixel/modem/modem_config/proto:checked_combo_py_pb2",
"//wireless/android/pixel/modem/modem_config/proto:combo_configuration_py_pb2",
"//wireless/android/pixel/modem/modem_config/proto:combo_py_pb2",
"//wireless/android/pixel/modem/modem_config/proto:pixel_project_py_pb2",
"//wireless/android/pixel/modem/modem_config/proto:rpc_py_pb2",
"//wireless/android/pixel/modem/modem_config/proto:specs_requirements_py_pb2",
],
)
pytype_strict_binary(
name = "rf_ca_config_generator_main",
srcs = ["rf_ca_config_generator_main.py"],
deps = [
":rf_ca_config_generator",
"//devtools/production/pyspanner",
"//file/colossus/public:cns",
"//pyglib:gfile",
"//third_party/py/absl:app",
],
)

Trace Codes

- **rf_ca_config_generator.py**
  - **main file called by**
    - ***rf_ca_config_generator_main***
- **rf_ca_config_generator_main.py**
- **rf_ca_config_generator_test.py**

rf_ca_config_generator_main.py

- **Main entry point for POR generation script**
- **Define the schema using the Protocol Buffers language (.proto file) in Enum struct**
- **Hard-coded *PixelProjectName*, *SkuName*, *DeviceVersion*, *PixelRelease* information to input into script in self-defined structure proto**

rf_ca_config_generator.py

- **Outputting *ca_configs.h* , *ca_configs.cpp* , *{project}_{sku}.cpp* files**
- **Implementing conversion CA Config Naming rules from**

go/pmw-ti-rfsw-por-gen-dd

2. ca_configs.h

```
with gfile.Open(os.path.join(device_dir, 'ca_configs.h'), 'w') as f:
      f.write(_CONFIGS_H_HEADER)
      for name in sorted(total_ca_configs_by_variable_name):
        f.write(f'extern const CaConfig_t {name};\n')
      f.write(_CONFIGS_H_TAIL)
```

```
def generate_ca_configs(
    self,
    device_name: str,
    projects: Collection[pixel_project_pb2.PixelProjectName],
    output_dir: str,
) -> None:
  """Generates CA configs code.

  It generates total ca configs c struct code and support ca configs per SKU.

  Args:
    device_name: The device name.
    projects: The projects of the device.
    output_dir: The output directory for generated files.
  """
  device_dir = os.path.join(output_dir, device_name)
  gfile.MakeDirs(device_dir)

  support_ca_configs_code_by_sku = (
      self.generate_support_ca_configs_code_of_sku_from_projects(projects)
  )
  # Collects all ca configs from all SKUs, we need to put them in a cpp file.
  total_ca_configs_by_variable_name: dict[str, str] = {}
  for sku, ca_config_code in support_ca_configs_code_by_sku.items():
    total_ca_configs_by_variable_name.update(
        ca_config_code.ca_config_by_variable_name
    )
    # with gfile.Open(os.path.join(device_dir, f'{sku}.cpp'), 'w') as f:
    #   f.write(_SUPPORT_C_HEADER)
    #   f.write(ca_config_code.support_ca_configs)

  with gfile.Open(os.path.join(device_dir, 'ca_configs.h'), 'w') as f:
    f.write(_CONFIGS_H_HEADER)
    for name in sorted(total_ca_configs_by_variable_name):
      f.write(f'extern const CaConfig_t {name};\n')
    f.write(_CONFIGS_H_TAIL)

  # with gfile.Open(os.path.join(device_dir, 'ca_configs.cpp'), 'w') as f:
  #   f.write(_CONFIGS_C_HEADER)
  #   for name in sorted(total_ca_configs_by_variable_name):
  #     f.write(total_ca_configs_by_variable_name[name])

  print(
      f'Use the following command to get generated files for {device_name}.'
  )
  print(f'fileutil cp {device_dir}/* .')
```

→ **output file**

```
#ifndef __CA_CONFIGS_H__
#define __CA_CONFIGS_H__
extern  "C" {
#include "ca_config.h"
}
extern const CaConfig_t ca_12_25_p12;
extern const CaConfig_t ca_12_25f_p12;
extern const CaConfig_t ca_12_2_4_p12;
extern const CaConfig_t ca_12_2_66_p12;
extern const CaConfig_t ca_12_2_7_p12;
extern const CaConfig_t ca_12_2_p12;
extern const CaConfig_t ca_12_2f_30_p12;
extern const CaConfig_t ca_12_2f_30f_66f_p12;
extern const CaConfig_t ca_12_2f_30f_p12;
extern const CaConfig_t ca_12_2f_4f_30f_p12;
extern const CaConfig_t ca_12_2f_4f_7f_p12;
extern const CaConfig_t ca_12_2f_4f_p12;
extern const CaConfig_t ca_12_2f_66f_p12;
extern const CaConfig_t ca_12_2f_7_p12;
extern const CaConfig_t ca_12_2f_7f_66f_p12;
extern const CaConfig_t ca_12_2f_7f_p12;
extern const CaConfig_t ca_12_2f_p12;
:
:
:
:
extern const CaConfig_t dc_n77f_n261_n2f_pn77_pn261;
extern const CaConfig_t dc_n77f_n261_n5_pn77_pn261;
extern const CaConfig_t dc_n77f_n261_n66f_pn77_pn261;
extern const CaConfig_t dc_n77f_n261_pn77_pn261;
extern const CaConfig_t dc_n78f_n258_pn78_pn258;

#endif
```

gTech
<users & products/>

After searching in database,
the Result is in

support_ca_configs_code_by_sku.items()

TYPE Mapping[str, RfCaConfigCode]

sku

TYPE

<class str>

c10_mmw

.
.
.
.

ca_config_code

TYPE

<class 'google3.wireless.android.
RfCaConfigCode'>

```
114  @dataclasses.dataclass(frozen=True)
115  class RfCaConfigCode:
116    """A code for RF CA configs.
117
118    Attributes:
119      ca_config_by_variable_name: A mapping from variable name to cpp struct
120        content of CA config.
121      support_ca_configs: A cpp vector of support CA configs of the sku.
122    """
123
124    ca_config_by_variable_name: Mapping[str, str]
125    support_ca_configs: str
```

generator.

ca_config_code.ca_config_by_variable_name

TYPE

<class 'dict'>

Collecting all keys

First key:

dc_66f_2f_13_n66f_n77f_p66_pn66

First value: TYPE <dataclasses RfCaConfigCode>

```
const CaConfig_t dc_66f_2f_13_n66f_n77f_p66_pn66 = {
    .name = "dc_66f_2f_13_n66f_n77f_p66_pn66",
    .band_cnt = 5,
    .bandConfigList = {
        {RF_BAND_66, true, true},
        {RF_BAND_2, true, false},
        {RF_BAND_13, false, false},
        {RF_BAND_NR_66, true, true},
        {RF_BAND_NR_77, true, false},
    },
};
```

ca_config_code.support_ca_configs

TYPE

<class str>

```
_SUPPORT_CA_CONFIG_OF_SKU = """
// CL: cl/{cl_id}
std::vector<const CaConfig_t *> {project_sku_name}_ca_configs {left_bracket}
{ca_config_name_list}
{right_bracket};
"""
```

```
// CL: cl/547655429
std::vector<const CaConfig_t *> l10_mmw_ca_configs {
    &ca_12_25f_p12, // CA_12A-25A
    &ca_12_2f_30_p12, // CA_2A-12A-30A, CA_2A-2A-12A-30A
            .
            .
            .
    &ca_12_2f_7_p12, // CA_2A-2A-7A-12A, CA_2A-7A-12A
    &dc_n48f_n261_pn48_pn261, // DC_n48A-n261A-n261A-n261A
};
```

3. ca_configs.cpp

```
with gfile.Open(os.path.join(device_dir, 'ca_configs.cpp'), 'w') as f:
  f.write(_CONFIGS_C_HEADER)
  for name in sorted(total_ca_configs_by_variable_name):
    f.write(total_ca_configs_by_variable_name[name])
```

```
def generate_ca_configs(
    self,
    device_name: str,
    projects: Collection[pixel_project_pb2.PixelProjectName],
    output_dir: str,
) -> None:
  """Generates CA configs code.

  It generates total ca configs c struct code and support ca configs per SKU.

  Args:
    device_name: The device name.
    projects: The projects of the device.
    output_dir: The output directory for generated files.
  """
  device_dir = os.path.join(output_dir, device_name)
  gfile.MakeDirs(device_dir)

  support_ca_configs_code_by_sku = (
      self.generate_support_ca_configs_code_of_sku_from_projects(projects)
  )
  # Collects all ca configs from all SKUs, we need to put them in a cpp file.
  total_ca_configs_by_variable_name: dict[str, str] = {}
  for sku, ca_config_code in support_ca_configs_code_by_sku.items():
    total_ca_configs_by_variable_name.update(
        ca_config_code.ca_config_by_variable_name
    )
    # with gfile.Open(os.path.join(device_dir, f'{sku}.cpp'), 'w') as f:
    #   f.write(_SUPPORT_C_HEADER)
    #   f.write(ca_config_code.support_ca_configs)

  # with gfile.Open(os.path.join(device_dir, 'ca_configs.h'), 'w') as f:
  #   f.write(_CONFIGS_H_HEADER)
  #   for name in sorted(total_ca_configs_by_variable_name):
  #     f.write(f'extern const CaConfig_t {name};\n')
  #   f.write(_CONFIGS_H_TAIL)

  with gfile.Open(os.path.join(device_dir, 'ca_configs.cpp'), 'w') as f:
    f.write(_CONFIGS_C_HEADER)
    for name in sorted(total_ca_configs_by_variable_name):
      f.write(total_ca_configs_by_variable_name[name])

  print(
      f'Use the following command to get generated files for {device_name}.'
  )
  print(f'fileutil cp {device_dir}/* .')
```

→ output file

```
extern  "C" {
#include "ca_configs.h"
}

const CaConfig_t ca_12_25_p12 = {
    .name = "ca_12_25_p12",
    .band_cnt = 2,
    .bandConfigList = {
        {RF_BAND_12, false, true},
        {RF_BAND_25, false, false},
    },
};

const CaConfig_t ca_12_25f_p12 = {
    .name = "ca_12_25f_p12",
    .band_cnt = 2,
    .bandConfigList = {
        {RF_BAND_12, false, true},
        {RF_BAND_25, true, false},
    },
};

const CaConfig_t ca_12_2_4_p12 = {
    .name = "ca_12_2_4_p12",
    .band_cnt = 3,
    .bandConfigList = {
        {RF_BAND_12, false, true},
        {RF_BAND_2, false, false},
        {RF_BAND_4, false, false},
    },
};

const CaConfig_t ca_12_2_66_p12 = {
    .name = "ca_12_2_66_p12",
    .band_cnt = 3,
    .bandConfigList = {
        {RF_BAND_12, false, true},
        {RF_BAND_2, false, false},
```

```
:
:
:
const CaConfig_t dc_n77f_n261_pn77_pn261 = {
    .name = "dc_n77f_n261_pn77_pn261",
    .band_cnt = 2,
    .bandConfigList = {
        {RF_BAND_NR_77, true, true},
        {RF_BAND_NR_257_261, false, true},
    },
};

const CaConfig_t dc_n78f_n258_pn78_pn258 = {
    .name = "dc_n78f_n258_pn78_pn258",
    .band_cnt = 2,
    .bandConfigList = {
        {RF_BAND_NR_78, true, true},
        {RF_BAND_NR_258, false, true},
    },
};
```

gTech
<users & products/>

After searching in database,
the Result is in

support_ca_configs_code_by_sku.items()

TYPE Mapping[str, RfCaConfigCode]

sku

TYPE

<class str>

c10_mmw

.

.

.

.

ca_config_code

TYPE

<class 'google3.wireless.android.
RfCaConfigCode'>

```
114  @dataclasses.dataclass(frozen=True)
115  class RfCaConfigCode:
116    """A code for RF CA configs.
117
118    Attributes:
119      ca_config_by_variable_name: A mapping from variable name to cpp struct
120        content of CA config.
121      support_ca_configs: A cpp vector of support CA configs of the sku.
122    """
123
124    ca_config_by_variable_name: Mapping[str, str]
125    support_ca_configs: str
```

generator.

ca_config_code.ca_config_by_variable_name

TYPE

<class 'dict'>

First key:

dc_66f_2f_13_n66f_n77f_p66_pn66

First value: TYPE <dataclasses RfCaConfigCode>

```
const CaConfig_t dc_66f_2f_13_n66f_n77f_p66_pn66 = {
    .name = "dc_66f_2f_13_n66f_n77f_p66_pn66",
    .band_cnt = 5,
    .bandConfigList = {
        {RF_BAND_66, true, true},
        {RF_BAND_2, true, false},
        {RF_BAND_13, false, false},
        {RF_BAND_NR_66, true, true},
        {RF_BAND_NR_77, true, false},
    },
};
```

Collecting all values

ca_config_code.support_ca_configs

TYPE

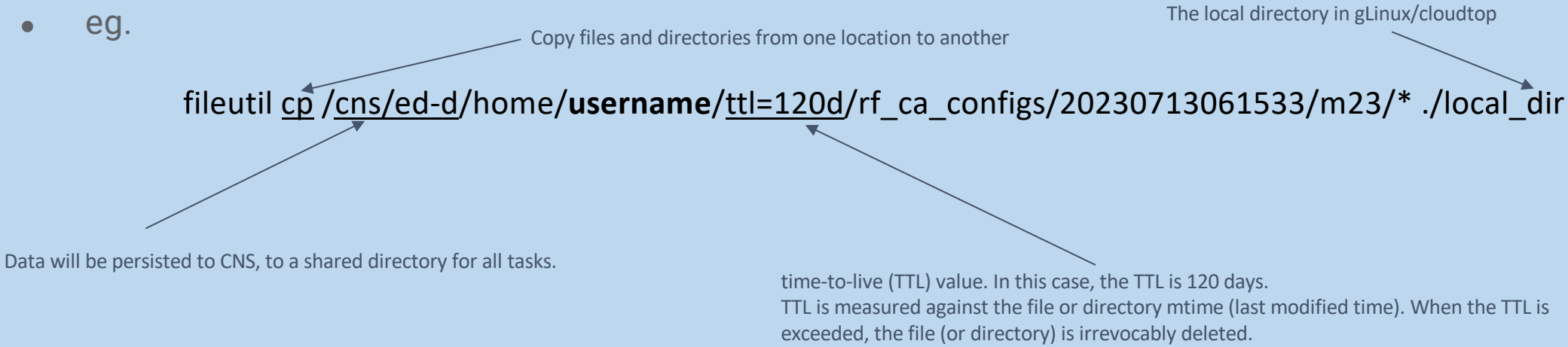
<class str>

```
_SUPPORT_CA_CONFIG_OF_SKU = """
// CL: cl/{cl_id}
std::vector<const CaConfig_t *> {project_sku_name}_ca_configs {left_bracket}
{ca_config_name_list}
{right_bracket};
"""
```

```
// CL: cl/547655429
std::vector<const CaConfig_t *> l10_mmw_ca_configs {
    &ca_12_25f_p12, // CA_12A-25A
    &ca_12_2f_30_p12, // CA_2A-12A-30A, CA_2A-2A-12A-30A
            .
            .
            .
    &ca_12_2f_7_p12, // CA_2A-2A-7A-12A, CA_2A-7A-12A
    &dc_n48f_n261_pn48_pn261, // DC_n48A-n261A-n261A-n261A
};
```

fileutil

- go/fileutil
- go/copy-tools
- `fileutil cp` is a command-line tool for copying a few files from place A to place B. It behaves like the UNIX cp command with the Google infrastructure and allows you to copy between corp and production, in both directions. For example, use `fileutil cp` to copy data out of Colossus to your local disk.
- fileutil cp <local file> <cns address>
- eg.



```
linyiyang@linyiyang:/google/src/cloud/linyiyang/Test2/google3/wireless/android/pixel/modem/ca_combo_gen/ca_combo_gen/rf_ca_config_gen$ fileutil ls /cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230629065658
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230629072102
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230629082525
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704063800
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704063901
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704064240
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704064711
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704065134
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704065529
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704070210
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704071126
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704071757
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704072045
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704072338
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704072605
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704073029
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704073205
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704073508
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704073637
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704073757
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704073920
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704121244
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704121419
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230704122838
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230705013143
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230705015042
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230705052725
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230705053019
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230705053439
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230705053626
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230705054128
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230705120011
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230706001004
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230706011126
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230706082106
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230706083434
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230707081421
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230707085310
/cns/ed-d/home/linyiyang/ttl=120d/rf_ca_configs/20230710095517
```

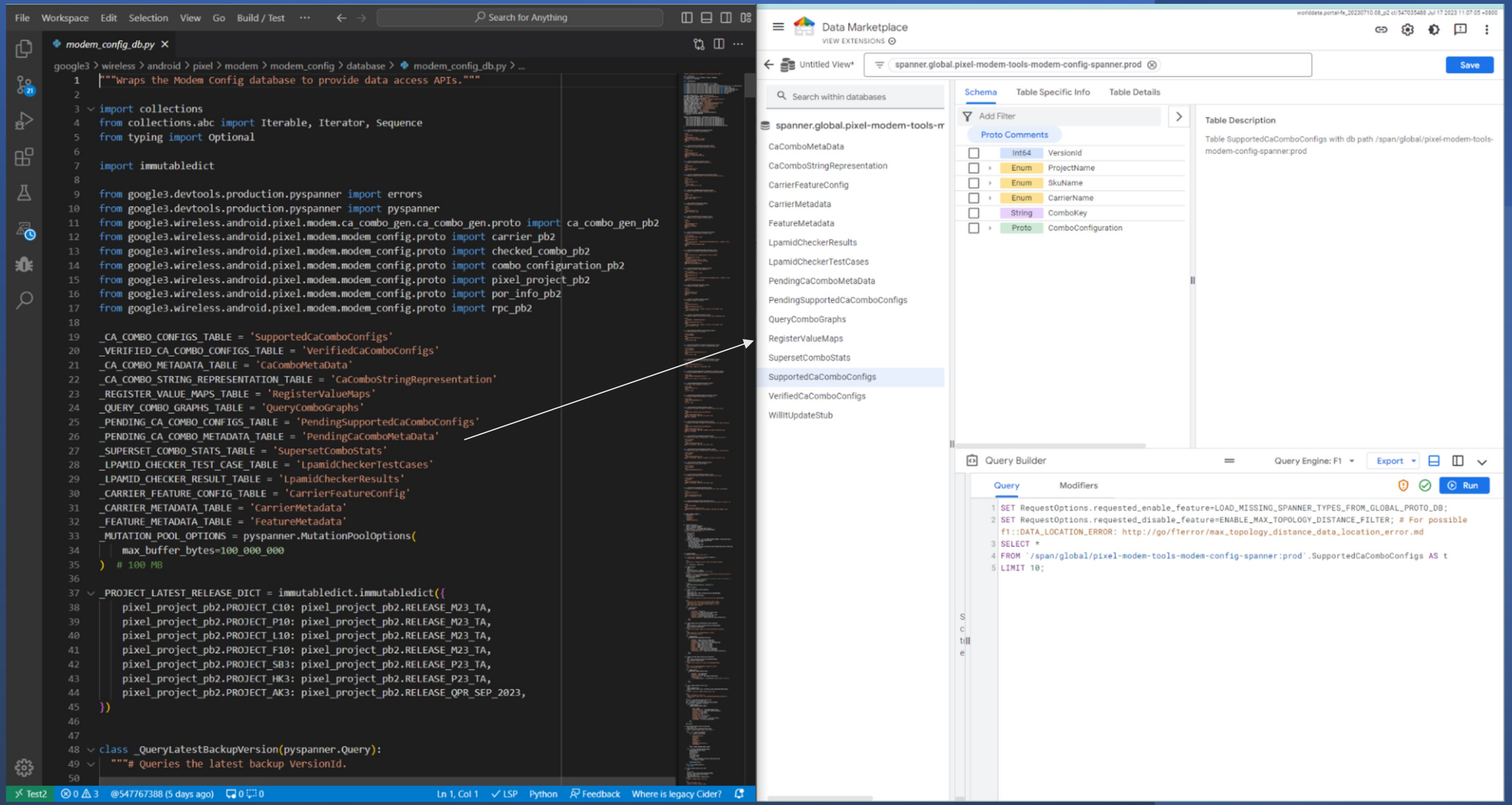
Colossus

- go/colossus-overview
- /cns/ed-d/
  - Providing a durable (but not permanent, because clusters turn down) repository for large amounts of unstructured data, such as videos.
  - Providing temporary storage of data in large workflow systems, for example log files.

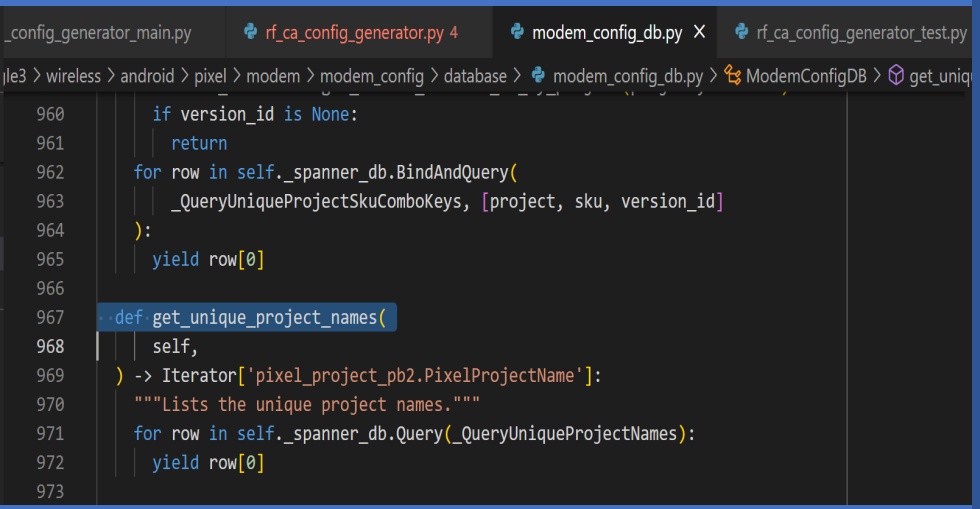
Database
Spanner

- Google’s globally distributed database
- go/spanner-codelab-python

Source: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis non erat sem


File Workspace Edit Selection View Go Build / Test
Search for Anything
modem_config_db.py
google3 > wireless > android > pixel > modem > modem_config > database > modem_config_db.py > ...
"""Wraps the Modem Config database to provide data access APIs."""
import collections
from collections.abc import Iterable, Iterator, Sequence
from typing import Optional
import immutabledict
from google3.devtools.production.pyspanner import errors
from google3.devtools.production.pyspanner import pyspanner
from google3.wireless.android.pixel.modem.ca_combo_gen.ca_combo_gen.proto import ca_combo_gen_pb2
from google3.wireless.android.pixel.modem.modem_config.proto import carrier_pb2
from google3.wireless.android.pixel.modem.modem_config.proto import checked_combo_pb2
from google3.wireless.android.pixel.modem.modem_config.proto import combo_configuration_pb2
from google3.wireless.android.pixel.modem.modem_config.proto import pixel_project_pb2
from google3.wireless.android.pixel.modem.modem_config.proto import por_info_pb2
from google3.wireless.android.pixel.modem.modem_config.proto import rpc_pb2
_CA_COMBO_CONFIGS_TABLE = 'SupportedCaComboConfigs'
_VERIFIED_CA_COMBO_CONFIGS_TABLE = 'VerifiedCaComboConfigs'
_CA_COMBO_METADATA_TABLE = 'CaComboMetaData'
_CA_COMBO_STRING_REPRESENTATION_TABLE = 'CaComboStringRepresentation'
_REGISTER_VALUE_MAPS_TABLE = 'RegisterValueMaps'
_QUERY_COMBO_GRAPHS_TABLE = 'QueryComboGraphs'
_PENDING_CA_COMBO_CONFIGS_TABLE = 'PendingSupportedCaComboConfigs'
_PENDING_CA_COMBO_METADATA_TABLE = 'PendingCaComboMetaData'
_SUPERSET_COMBO_STATS_TABLE = 'SupersetComboStats'
_LPAMID_CHECKER_TEST_CASE_TABLE = 'LpamidCheckerTestCases'
_LPAMID_CHECKER_RESULT_TABLE = 'LpamidCheckerResults'
_CARRIER_FEATURE_CONFIG_TABLE = 'CarrierFeatureConfig'
_CARRIER_METADATA_TABLE = 'CarrierMetadata'
_FEATURE_METADATA_TABLE = 'FeatureMetadata'
_MUTATION_POOL_OPTIONS = pyspanner.MutationPoolOptions(
max_buffer_bytes=100_000_000
) # 100 MB
_PROJECT_LATEST_RELEASE_DICT = immutabledict.immutabledict({
pixel_project_pb2.PROJECT_C10: pixel_project_pb2.RELEASE_M23_TA,
pixel_project_pb2.PROJECT_P10: pixel_project_pb2.RELEASE_M23_TA,
pixel_project_pb2.PROJECT_L10: pixel_project_pb2.RELEASE_M23_TA,
pixel_project_pb2.PROJECT_F10: pixel_project_pb2.RELEASE_M23_TA,
pixel_project_pb2.PROJECT_SB3: pixel_project_pb2.RELEASE_P23_TA,
pixel_project_pb2.PROJECT_HK3: pixel_project_pb2.RELEASE_P23_TA,
pixel_project_pb2.PROJECT_AK3: pixel_project_pb2.RELEASE_QPR_SEP_2023,
})
class _QueryLatestBackupVersion(pyspanner.Query):
"""# Queries the latest backup VersionId.
Test2 @547767388 (5 days ago) Ln 1, Col 1 LSP Python Feedback Where is legacy Cider?
Data Marketplace
VIEW EXTENSIONS
Untitled View*
spanner.global.pixel-modem-tools-modem-config-spanner.prod
Save
Search within databases
spanner.global.pixel-modem-tools-m
CaComboMetaData
CaComboStringRepresentation
CarrierFeatureConfig
CarrierMetadata
FeatureMetadata
LpamidCheckerResults
LpamidCheckerTestCases
PendingCaComboMetaData
PendingSupportedCaComboConfigs
QueryComboGraphs
RegisterValueMaps
SupersetComboStats
SupportedCaComboConfigs
VerifiedCaComboConfigs
WillItUpdateStub
Schema
Table Specific Info
Table Details
Add Filter
Proto Comments
Int64 VersionId
Enum ProjectName
Enum SkuName
Enum CarrierName
String ComboKey
Proto ComboConfiguration
Table Description
Table SupportedCaComboConfigs with db path /span/global/pixel-modem-tools-modem-config-spanner:prod
Query Builder
Query Engine: F1
Export
Query
Modifiers
Run
SET RequestOptions.requested_enable_feature=LOAD_MISSING_SPANNER_TYPES_FROM_GLOBAL_PROTO_DB;
SET RequestOptions.requested_disable_feature=ENABLE_MAX_TOPOLOGY_DISTANCE_FILTER; # For possible f1::DATA_LOCATION_ERROR: http://go/f1error/max_topology_distance_data_location_error.md
SELECT *
FROM `/span/global/pixel-modem-tools-modem-config-spanner:prod`.SupportedCaComboConfigs AS t
LIMIT 10;

```
399       with gfile.Open(os.path.join(device_dir, 'ca_configs.h'), 'w') as f:
400         f.write(_CONFIGS_H_HEADER)
401         for name in sorted(total_ca_configs_by_variable_name):
402           f.write(f'extern const CaConfig_t {name};\n')
403         f.write(_CONFIGS_H_TAIL)
404
405       with gfile.Open(os.path.join(device_dir, 'ca_configs.cpp'), 'w') as f:
406         f.write(_CONFIGS_C_HEADER)
407         for name in sorted(total_ca_configs_by_variable_name):
408           f.write(total_ca_configs_by_variable_name[name])
409
410       print(
411           f'Use the following command to get generated files for {device_name}.'
412       )
413       print(f'fileutil cp {device_dir}/* .')
414
415     def generate_all_ca_configs(
416         self,
417         output_dir: str,
418     ) -> None:
419       """Generates CA configs code for all projects.
420
```

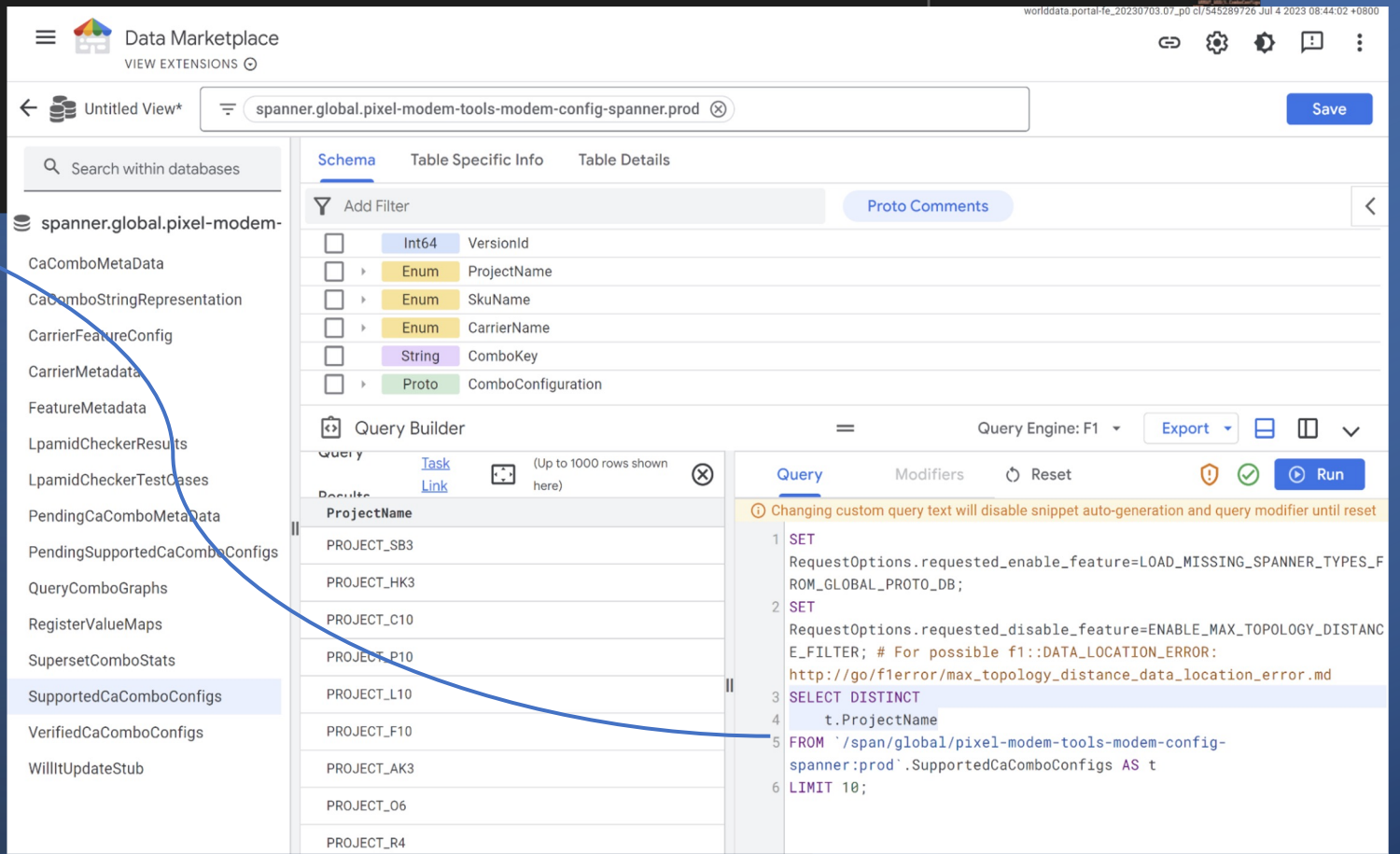


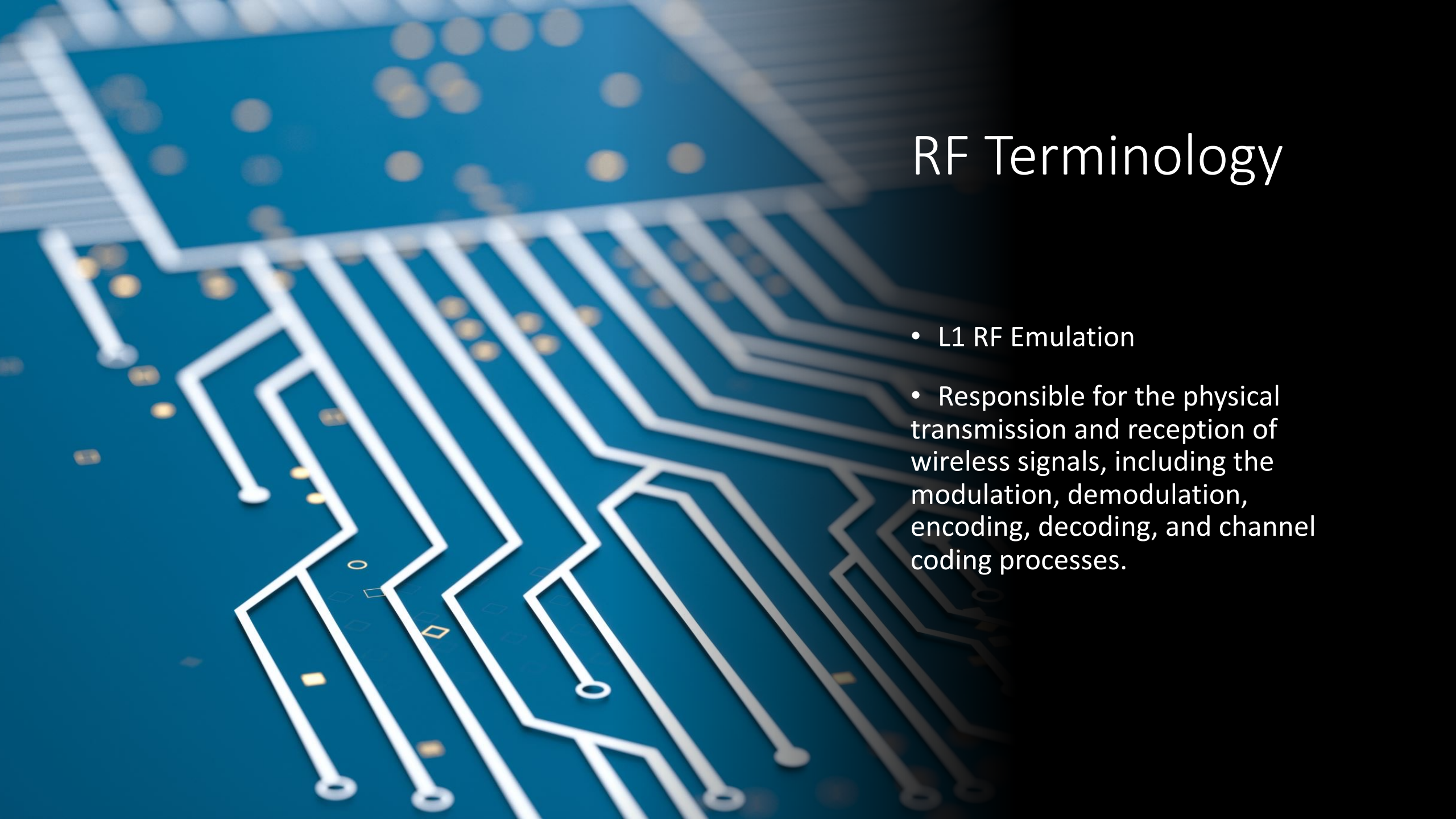

```python
def get_unique_project_names(
    self,
) -> Iterator['pixel_project_pb2.PixelProjectName']:
  """Lists the unique project names."""
  for row in self._spanner_db.Query(_QueryUniqueProjectNames):
    yield row[0]
```

modem_config_db.py

google3 > wireless > android > pixel > modem > modem_config > database > modem_config_db.py > _QueryUniqueProjectNames

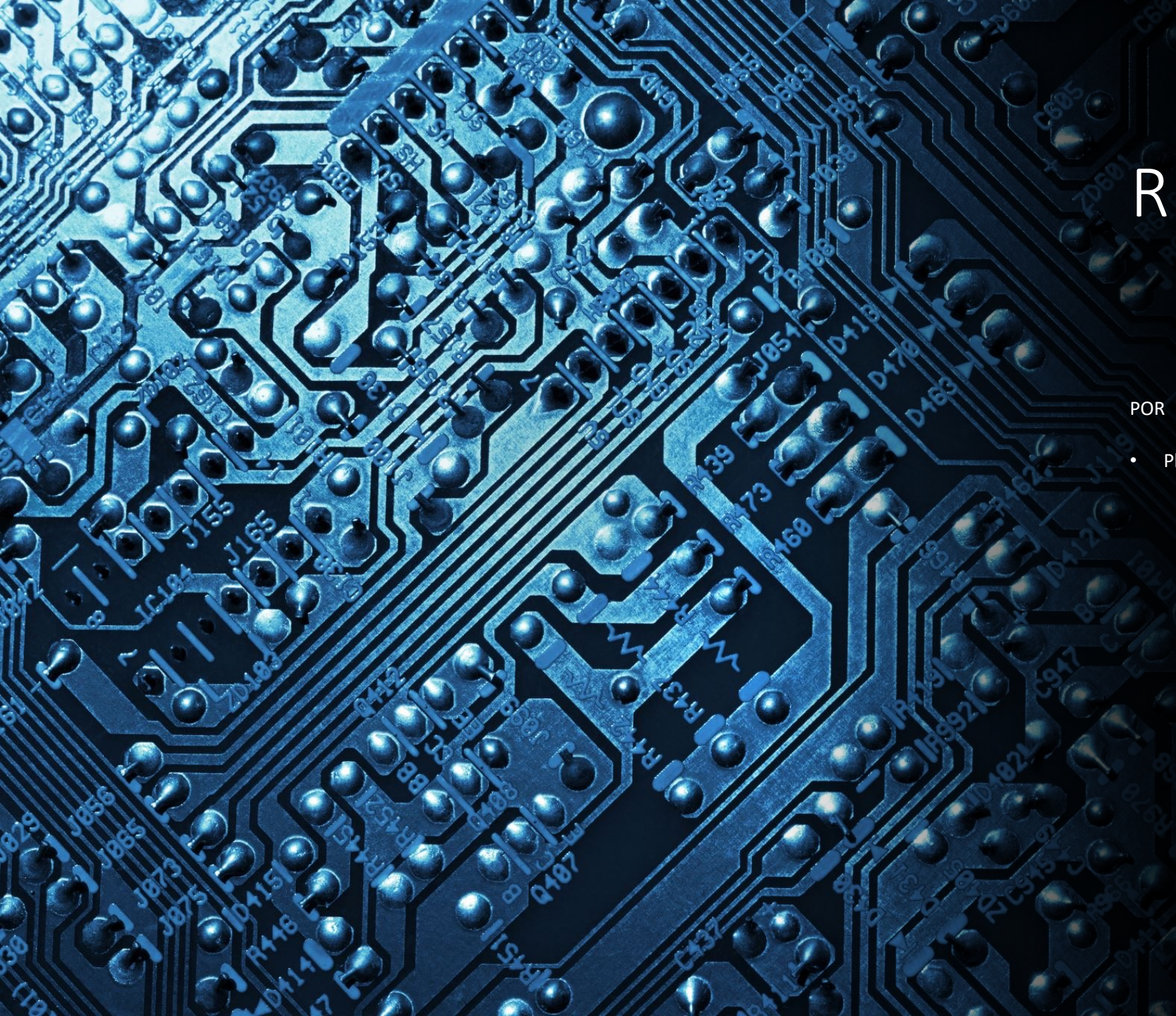
```
319
320 class _QueryUniqueProjectNames(pyspanner.Query):
321   """# Queries the unique project names.
322
323   SELECT DISTINCT
324     t.ProjectName
325   FROM
326     SupportedCaComboConfigs AS t
327   """
```





RF Terminology

- L1 RF Emulation
- Responsible for the physical transmission and reception of wireless signals, including the modulation, demodulation, encoding, decoding, and channel coding processes.



RF Terminology

POR

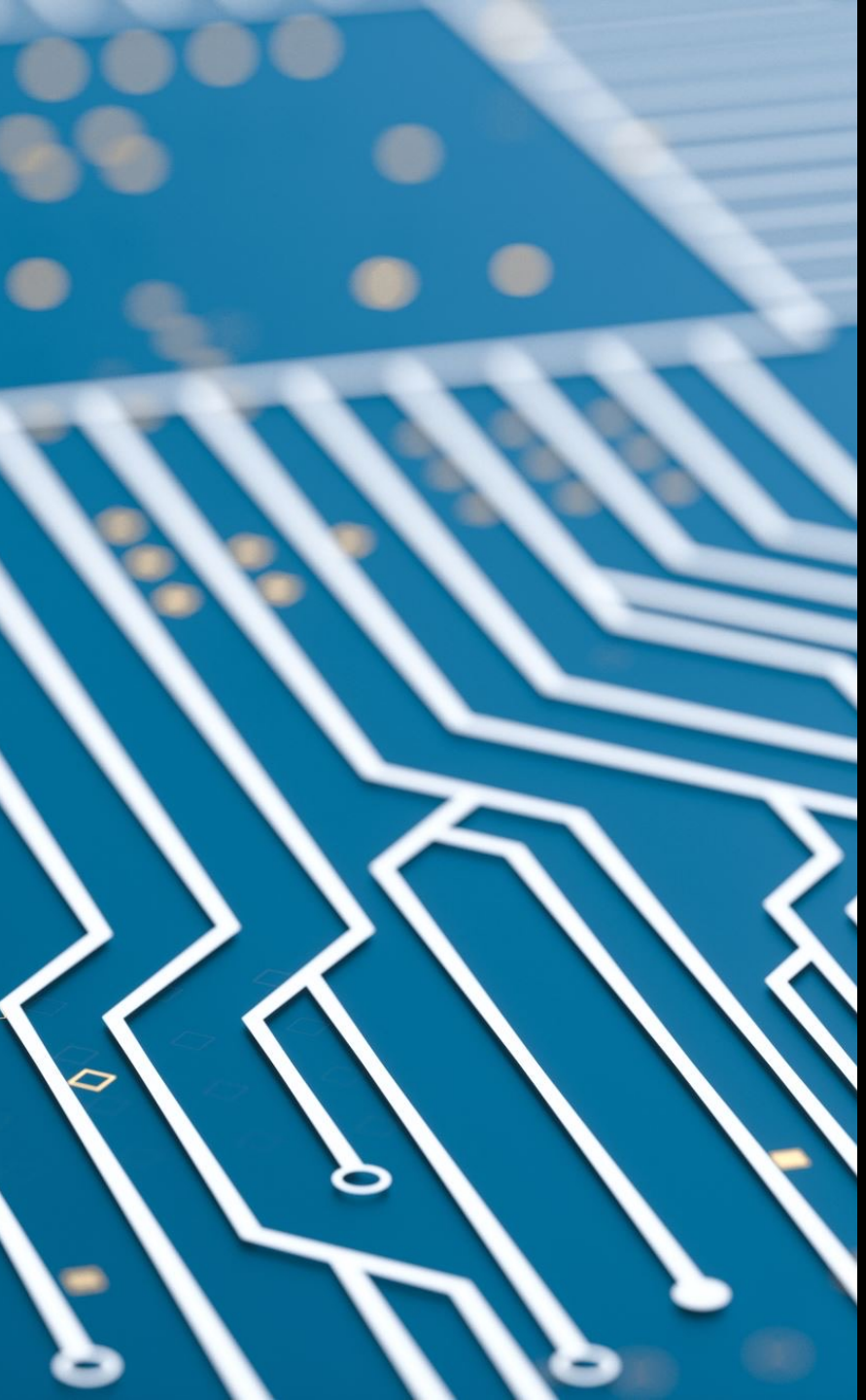
- Plan On Record

RF design to ensure the proper and reliable operation of RF devices and systems upon power-up monitor the power supply voltage and generate a reset signal to initiate the proper startup sequence

POR referring to a set of default values or configurations that can be used when specific parameters such as LTE bandwidth, EARFCN (E-UTRA Absolute Radio Frequency Channel Number), and NR-ARFCN (New Radio Absolute Radio Frequency Channel Number) are not specified. The reference to "36.101" and "38.104" suggests that these documents could provide additional details or guidelines for the default values.

RF Terminology

- E-ARFCN
  - E-UTRA Absolute Radio Frequency Channel Number
  - a parameter used in Long-Term Evolution (LTE) networks
    - Downlink EARFCN
    - Uplink EARFCN



RF Terminology

- 3GPP
  - 36.101 and 38.104
    - Provide guidance and specifications that help ensure the proper design, deployment, and operation of RF communication systems within these cellular networks.
    - Refer to specific technical specifications or standards used in the context of cellular networks, particularly for (3rd Generation Partnership Project) systems like LTE (Long-Term Evolution) and 5G (fifth generation) networks.
    - 3GPP is a collaboration between telecommunications standards organizations that develops global standards for mobile communications technologies, including GSM (Global System for Mobile Communications), 3G (Third Generation), and 4G (Fourth Generation) LTE (Long-Term Evolution) networks. Each release of the 3GPP specifications represents a significant milestone in the evolution of mobile communication technologies.

RF Terminology

- 36.101
  - Provides details about the User Equipment (UE) radio transmission and reception characteristics for LTE networks. It covers topics such as radio frequency bands, channel bandwidths, power control, modulation schemes, coding schemes, and radio performance requirements.
- 38.104
  - Focuses on the radio transmission and reception aspects of 5G NR (New Radio) networks. It provides detailed information about the physical layer procedures and protocols, including radio resource management, channel coding, modulation, and multiplexing techniques.
  - Covers topics such as carrier frequency ranges, channel bandwidths, multiple access schemes, link adaptation, beamforming, and performance requirements for 5G NR systems. It serves as a key reference for implementing 5G networks and ensuring consistent and reliable RF communication between 5G devices and base stations.

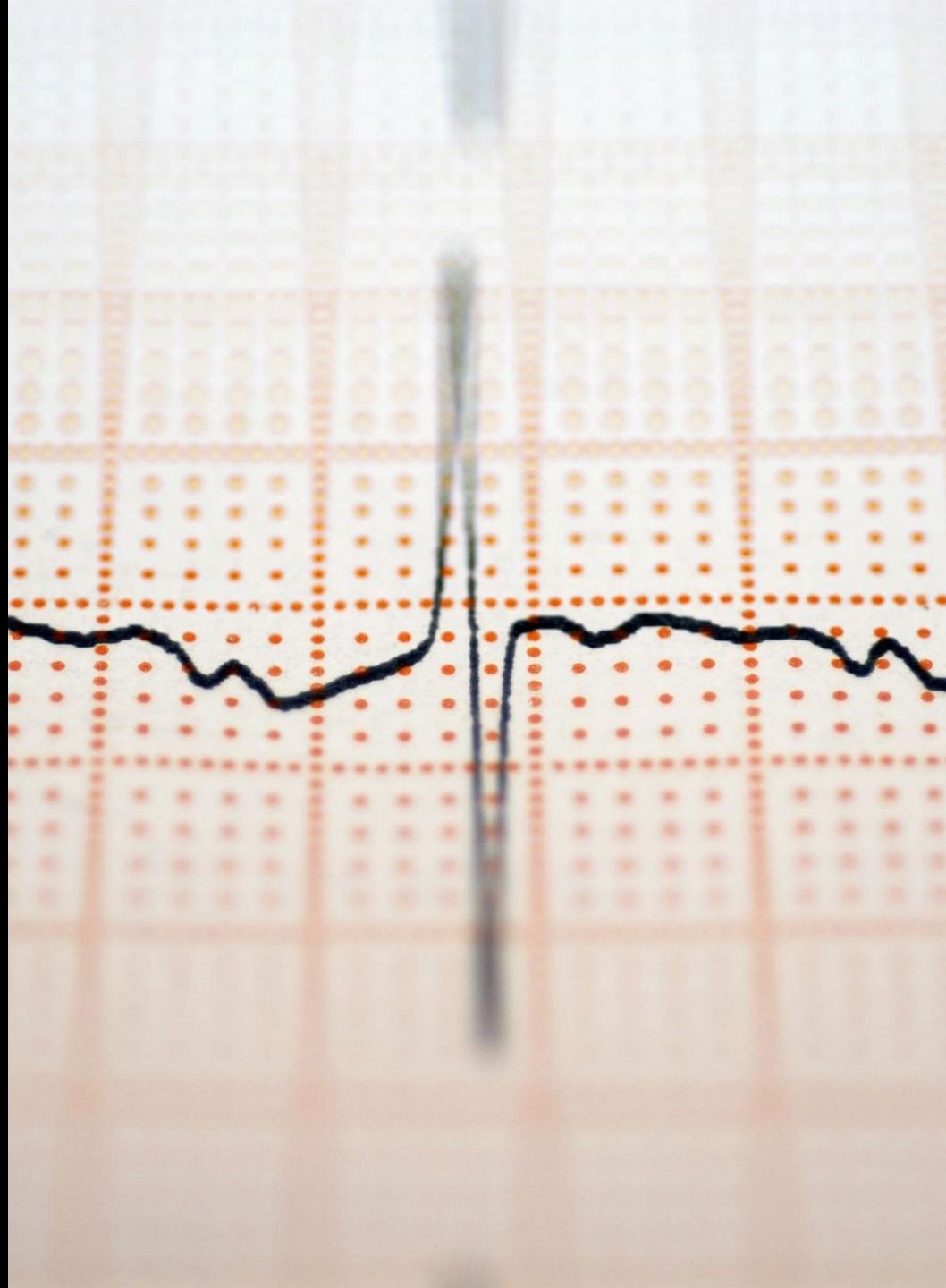
RF Terminology

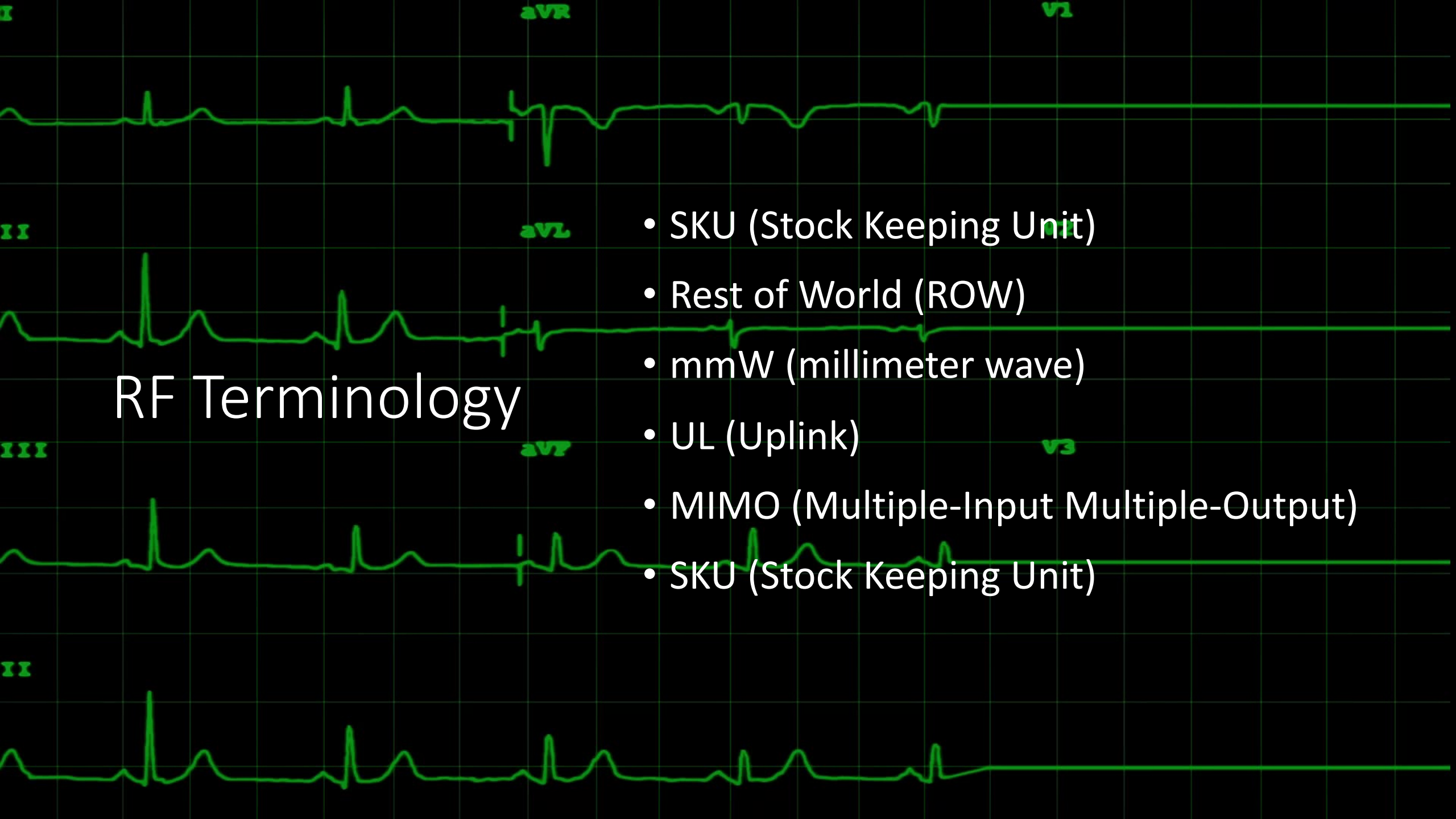
- L1 RF Emulation
- Responsible for the physical transmission and reception of wireless signals, including the modulation, demodulation, encoding, decoding, and channel coding processes.

RF Terminology

- ARFCN
  - used in 2G networks
- E-ARFCN
  - used in 4G and 5G network
  - The purpose of both parameters is to provide a standardized representation for radio frequency channels, allowing devices to tune into the correct frequency for communication.

It's important to note that the specific frequency associated with an ARFCN or E-ARFCN value depends on the cellular band being used, as different frequency bands are allocated for different regions and network deployments.





RF Terminology

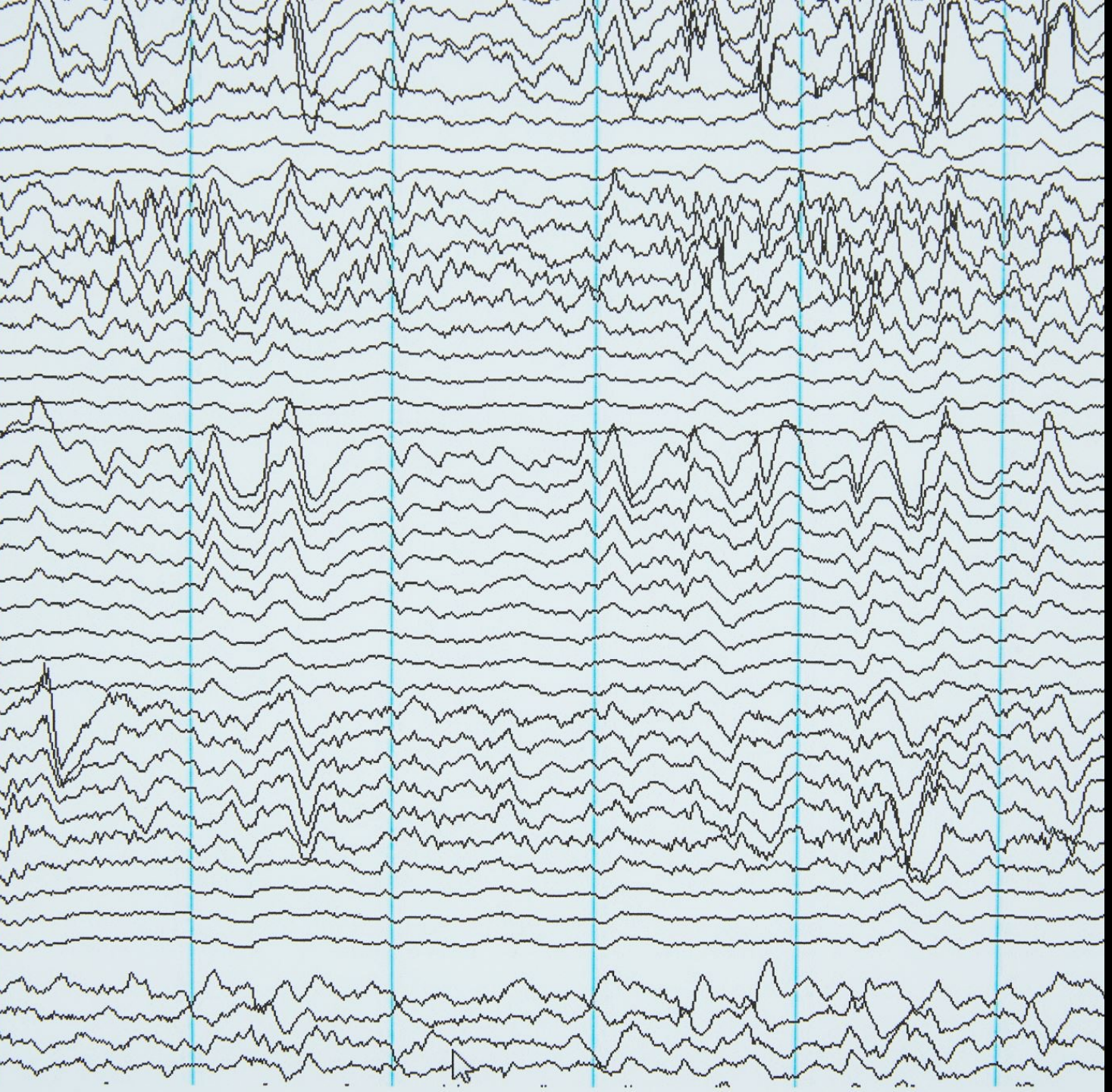
- SKU (Stock Keeping Unit)
- Rest of World (ROW)
- mmW (millimeter wave)
- UL (Uplink)
- MIMO (Multiple-Input Multiple-Output)
- SKU (Stock Keeping Unit)

RF Terminology

- US mmW SKU
  - Refers to a Stock Keeping Unit specifically tailored for the United States (US) market that utilizes mmW (millimeter wave) frequencies. Millimeter waves are high-frequency radio waves that offer significant bandwidth for faster data transmission. This SKU is optimized for the mmW frequency range and is typically used for high-speed, short-range wireless communication applications.

RF Terminology

- NA sub6 SKU
  - Stands for North America (NA) sub6 Stock Keeping Unit. "sub6" refers to frequencies below 6 GHz, which includes the traditional cellular frequency bands. This SKU is designed for wireless communication systems operating within the sub6 GHz range, which is commonly used for wider coverage and longer-range communications compared to mmW frequencies. It is typically used in applications such as cellular networks, Wi-Fi, and other wireless technologies.



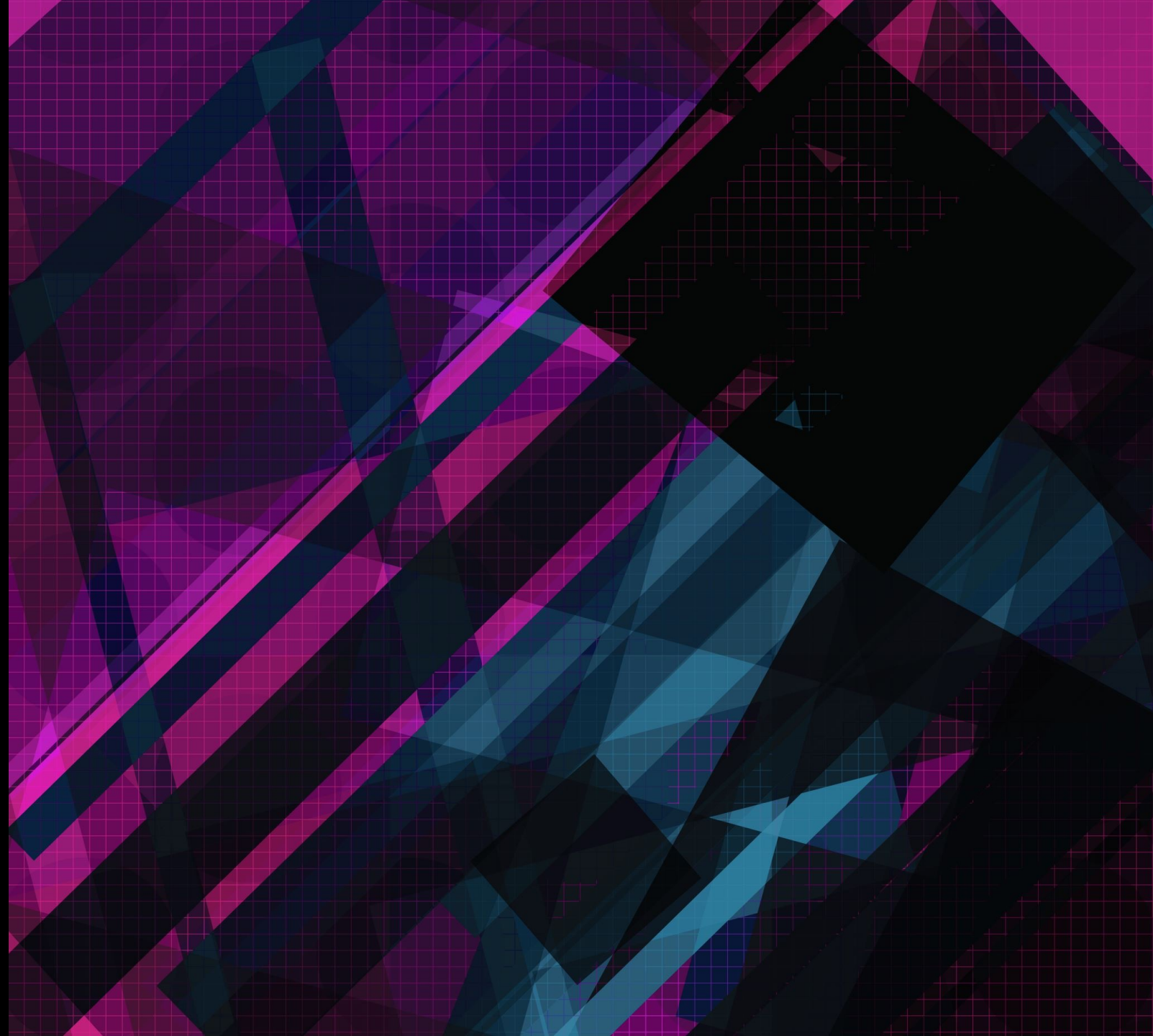
RF Terminology

- ROW sub6 SKU
  - Represents the Rest of World (ROW) sub6 Stock Keeping Unit. Similar to the NA sub6 SKU, it is optimized for frequencies below 6 GHz and is designed for wireless communication systems deployed in regions outside of North America. It is tailored to the specific requirements and regulations of various countries and regions around the world.

RF Terminology

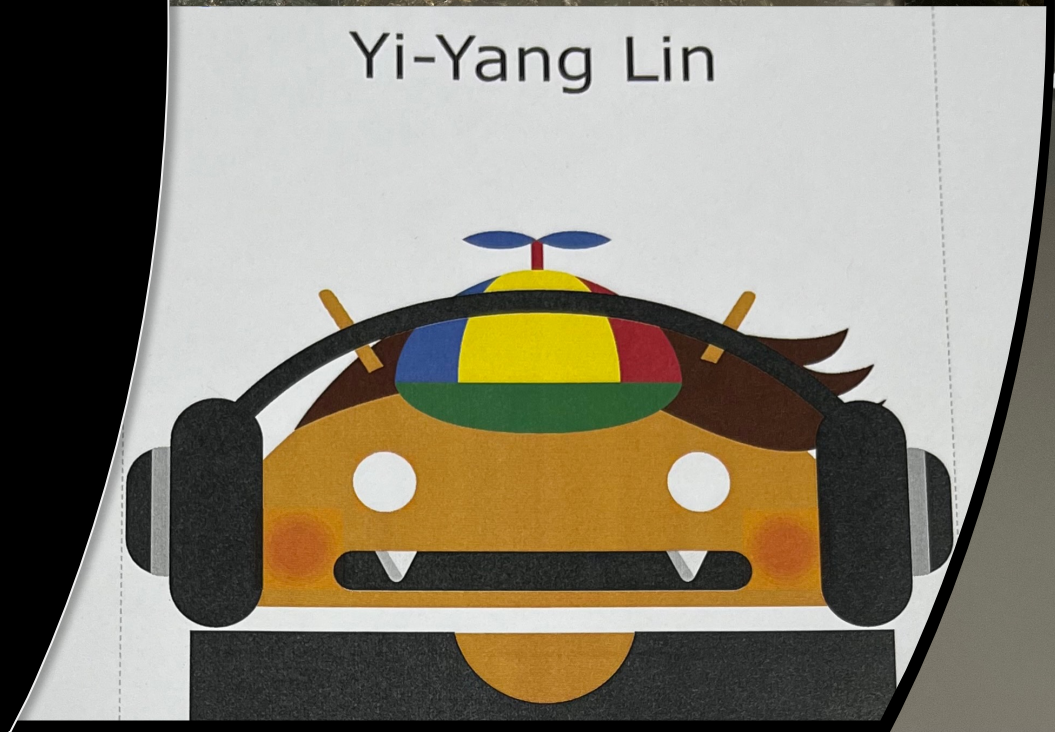
- SKU (Stock Keeping Unit)
- Rest of World (ROW)
- mmW (millimeter wave)
- UL (Uplink)
- MIMO (Multiple-Input Multiple-Output)
- SKU (Stock Keeping Unit)

Google Life



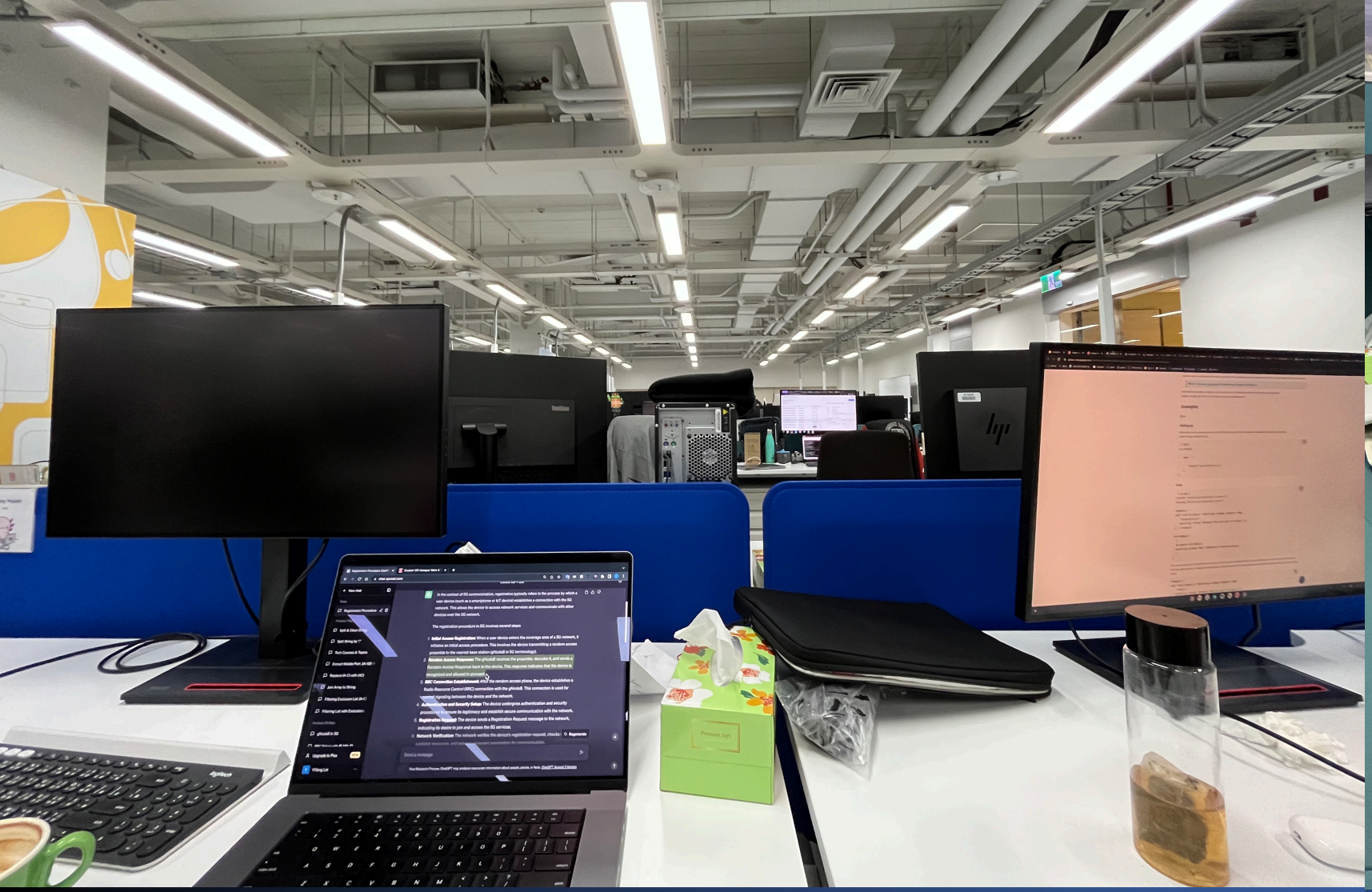
Onboarding


Google
Intern
Create
Design
Code
Build
for
everyone
Yi-Yang
Lin
Google


Yi-Yang Lin

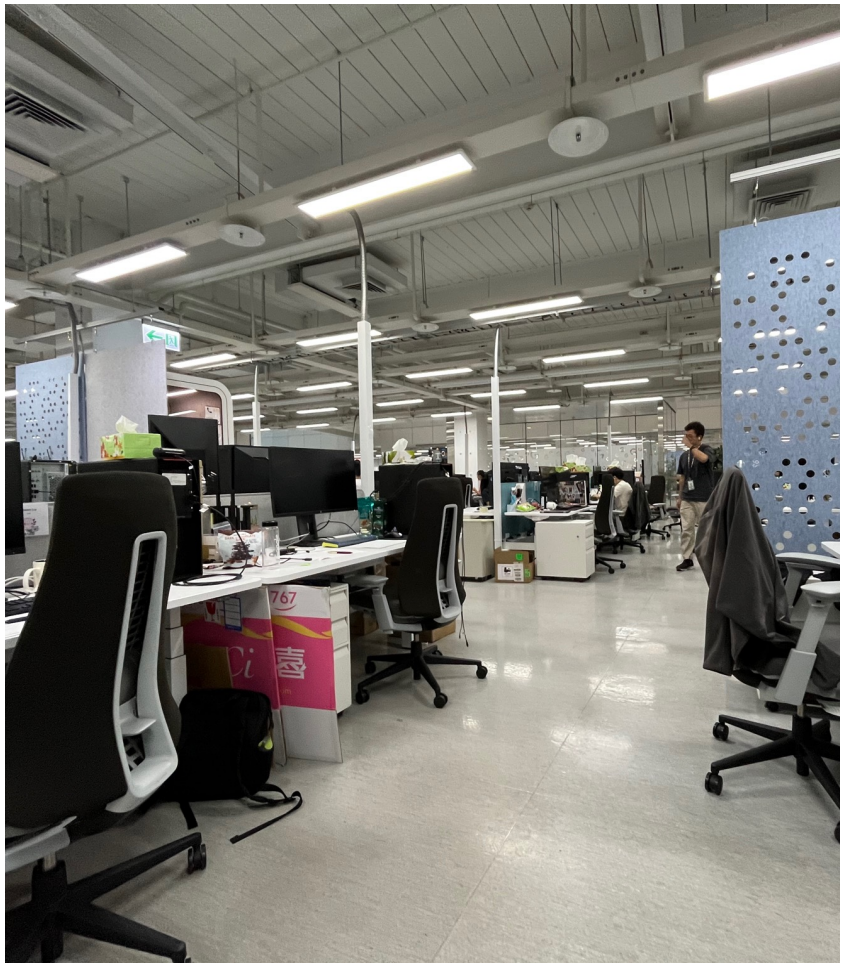

Meet me on GVC
100

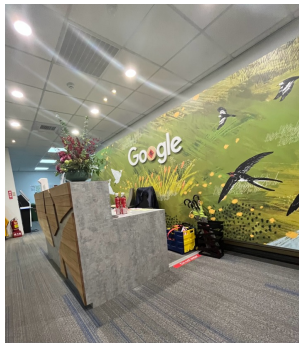
Work with Beer

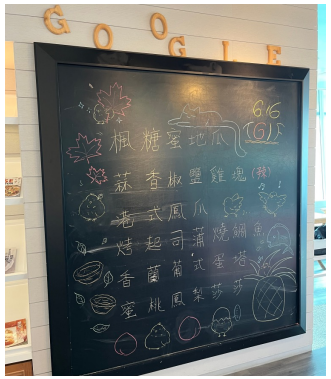


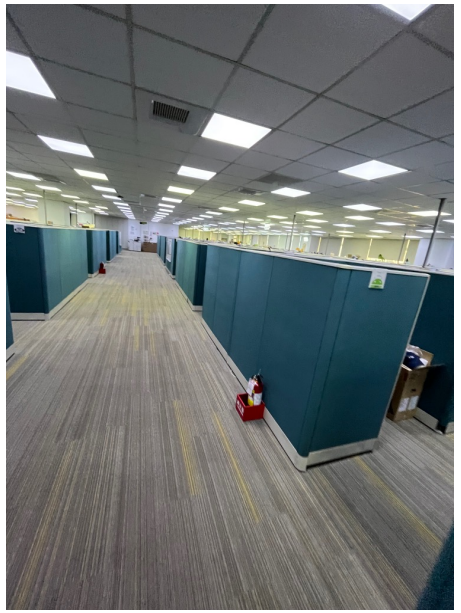
Study with ChatGPT


Google Pixel 7a
一拍驚艷，輕鬆入手




Google


GOOGLE

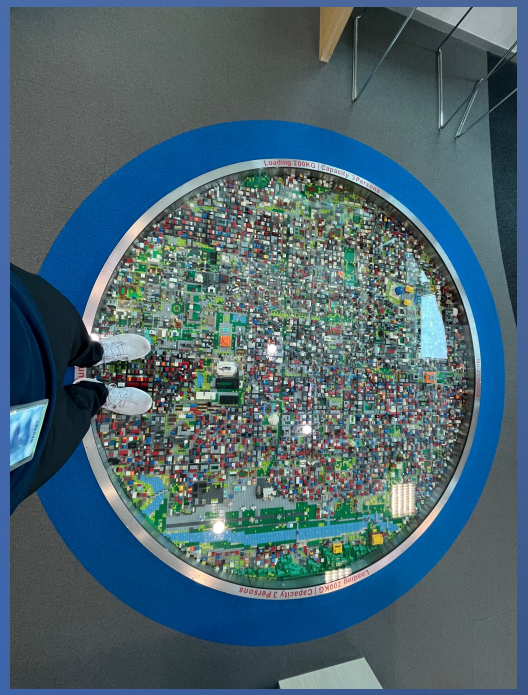


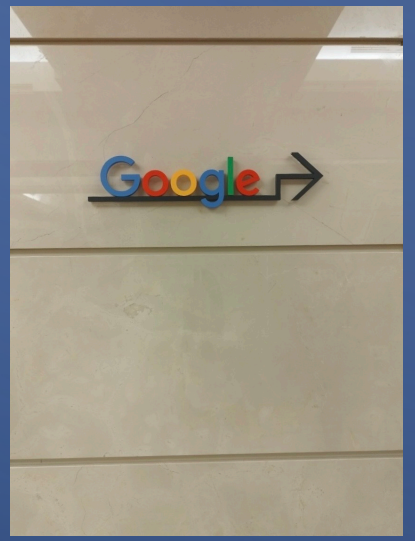

Hello
Noogler
New Amenities
Medical Station @ L6 Side A
GBus Schedule


Google

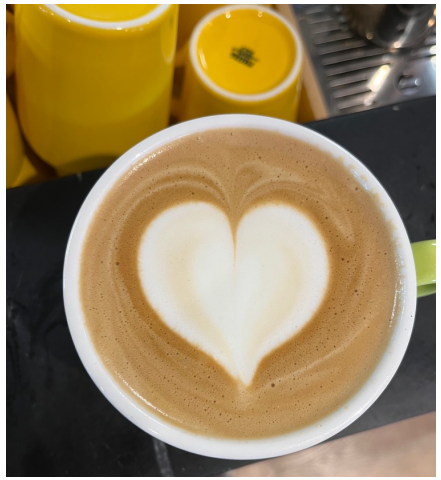

Google

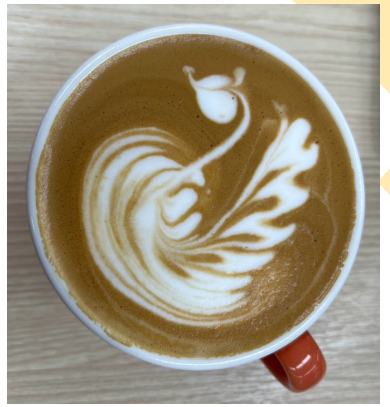

CLASSIC BLUE | LIVING CORAL
SERENITY | ROSE QUARTZ
16 CAFE MEETS HYDRATION POINT
15 MICROKITCHEN
14 COFFEE CORNER NAP ROOM
13 MICROKITCHEN
12 COFFEE CORNER NAP ROOM
11 MICROKITCHEN
10 WELLNESS CENTER MOTHER'S ROOM MAIL ROOM
TRANSFER FLOOR
9 CAFE ZAOKA | HONG TE
8 COFFEE CORNER
7 MICROKITCHEN
6 COFFEE CORNER
5 HYDRATION POINT
4 MICROKITCHEN
3 HYDRATION POINT
2 VISITOR LOBBY DEMO SPACE
TANGERINE TANGO
1 VENDOR LAB | MAKERSPACE FITNESS CENTER




Google

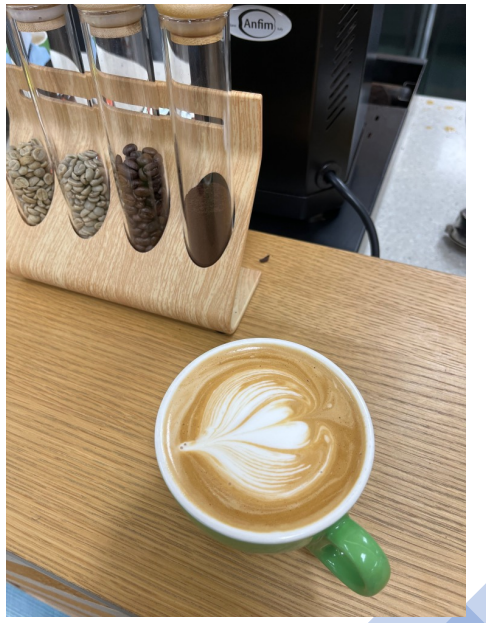

COFFEE BREWING METHODS





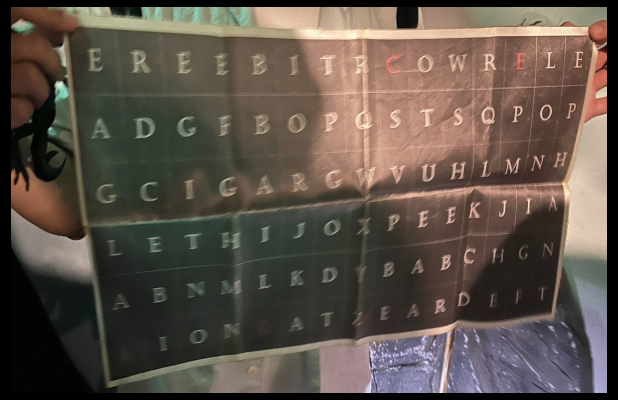

HONG TE


Häagen-Dazs


Anfim


Google

Team Build &
Conversation with Managers



2023 GCN Interns Escape Room Activity

Google Serve – Volunteering Opportunities


Terminal D1


SUMMER
LAUNCH PARTY
PRESS
START
GOOD
TO GO


BHKS @GOOGLE


Google


android
be together. not the same.


Google


Google


Google
Häagen-Dazs

TGIF

- Thursday Googler Information Forum
- Company-wide meeting


LAYA COMBO
MR. BURGER
RICE BURGER
LAYA LIGHT
BEVERAGE
TOAST
COFFEE


川燙
季節時蔬
餐具不交叉使用
避免單素餐點混味


百香果
Passion fruit
草莓
Strawberry


MICHY
No.1
$100 454 kcal
蒜香白酒蛤蜊麵
Clam Pasta With White Wine And Garlic
No.2
$115 365 kcal
全素風味蔬菜麵
Vegetable Pasta With Virgin Olive Oil
No.3
$100 650 kcal
奶油菇菇雞肉麵
Chicken And Comprehensive Mushroom Pasta With Cream Sauce
No.4
$100 778 kcal
青醬奶油培根麵
Butter Bacon Pasta With Green Sauce
No.5
$90 479 kcal
蕃茄肉醬麵
Bolognese Pasta
No.6
$100 678 kcal
蕃茄菇菇雞肉麵
Chicken And Comprehensive Mushroom Pasta With Tomato Sauce
No.7
$115 635 kcal
墨西哥蕃茄肉醬燉飯
Mexico Bolognese Risotto With Cream Sauce
No.8
$ 95/ 642.5 kcal
乾炒辣味雞肉麵
Spicy Chicken Pasta
No.9
$110 688 kcal
印式香料咖哩雞肉飯
Chicken Tikka Masala Curry Rice
No.11
No.12
鮮奶奶酪
Panna Cotta
69kcal $25
紅茶凍
Black Tea Jelly
69kcal $25
芒果青
Green Mango Sorbet
249.5kcal $50
如需外帶
請提前告知
$90 437 kcal
蘿蔓風味義式冷麵
Vegetable Cold Pasta With Virgin Olive Oil
蛋奶素
食用前請先拌勻
建議勿放至隔餐




TAIWAN BEER
BUSCH




Häagen-Dazs


Thank you