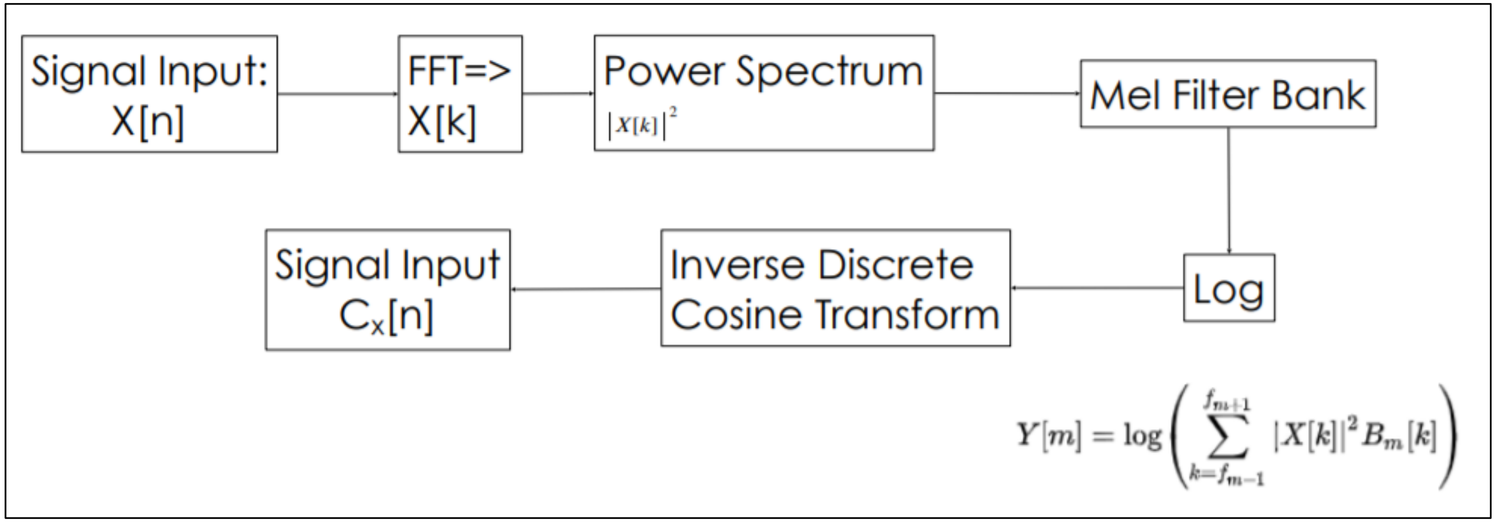
Accelerate calculation of mel-frequency cepstral coefficients by HLS

Team 3: 林承德, 張俊堉, 蔡孟桓

github.com/WingTillDie/Real-time_Respiratory_Monitor_MFCC

Preprocessing stage of Heroic-Faith Airmod

- Airmod of Heroic-Faith preprocesses data as mel-frequency cepstral coefficients (MFCCs), then feed into machine learning model
- We use high-level synthesis (HLS) to accelerate the computation of MFCCs



Signal Input: X[n]

FFT=> X[k]

Power Spectrum $|X[k]|^2$

Mel Filter Bank

Log

Inverse Discrete Cosine Transform

Signal Input $C_x[n]$

$$Y[m] = \log\left(\sum_{k=f_{m-1}}^{f_{m+1}} |X[k]|^2 B_m[k]\right)$$

System integration

- Airmod: Python, Host code: OpenCL
- Use Vitis, target platform: Alveo U50
- Processing steps:

1. Airmod write data to disk
2. Host code read data from disk
3. Process data by kernel
4. Write processed result to disk
5. Airmod read processed data from disk, then feed into machine learning model

Step 1: Airmod write data to disk

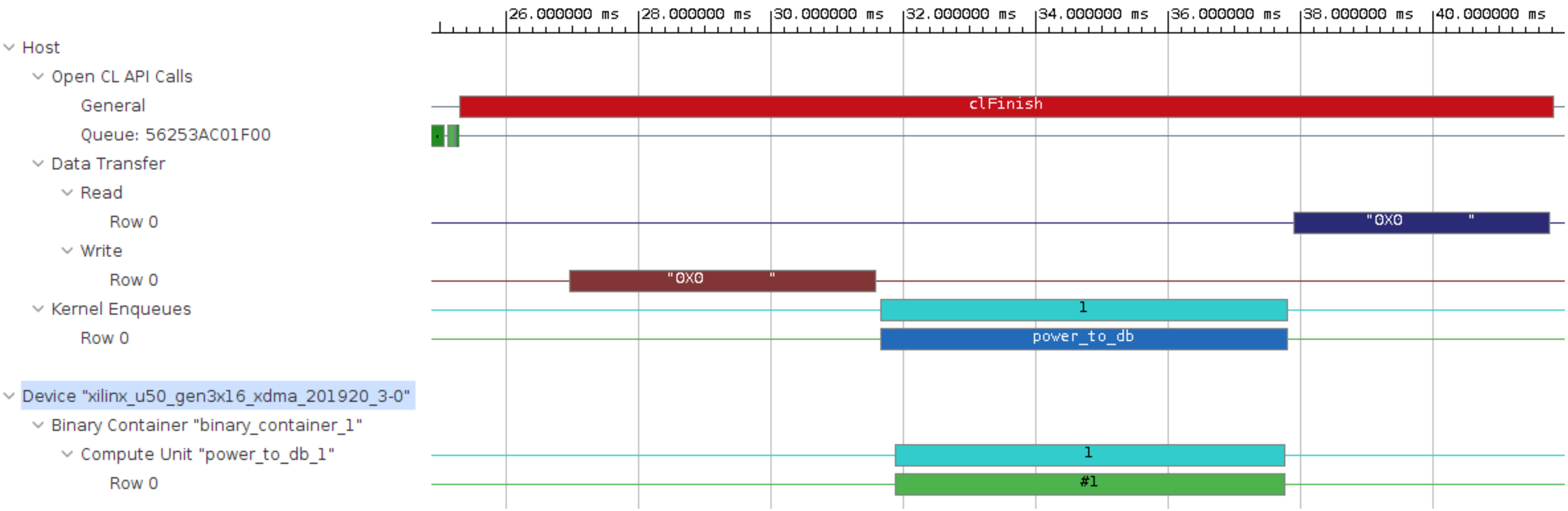
- Airmod write numpy ndarray to disk
- For example, P is array with shape (128, 938)
- `P.tofile("P.array")`
  - P.array is in binary format

```
array([[1.58494657e-06, 5.77005303e-07, 1.63309445e-09, ...,
        2.37664272e-08, 2.45290324e-05, 1.00442062e-03],
       [6.74685035e-07, 2.36331229e-07, 8.12508869e-07, ...,
        2.32557493e-07, 2.58855295e-05, 1.20690251e-03],
       [6.31601473e-05, 5.18658335e-05, 3.57480690e-05, ...,
        1.70830175e-05, 5.44760936e-05, 1.88787303e-03],
       ...,
       [5.58200052e-07, 1.34728246e-07, 6.29370041e-10, ...,
        5.67968658e-09, 4.10374181e-09, 1.34892459e-07],
       [5.14760178e-07, 1.35840369e-07, 1.37220771e-09, ...,
        9.52569933e-09, 1.23505759e-08, 1.93342999e-07],
       [4.91059820e-07, 1.20994118e-07, 2.61981242e-09, ...,
        3.46911983e-09, 7.87195200e-09, 1.84565499e-07]])
```

Step 2: host code read data from disk

```
void *ptr=nullptr;
posix_memalign(&ptr,4096,SIZE_BUF_power_to_db*sizeof(double))
BUF_power_to_db = reinterpret_cast<int*>(ptr);
FILE *fp = fopen("P.array", "rb");
fread(BUF_power_to_db, SIZE_BUF_power_to_db*sizeof(double), 1, fp);
fclose(fp);
```

Step 3: Process data by kernel



Step 4: host code write processed data to disk

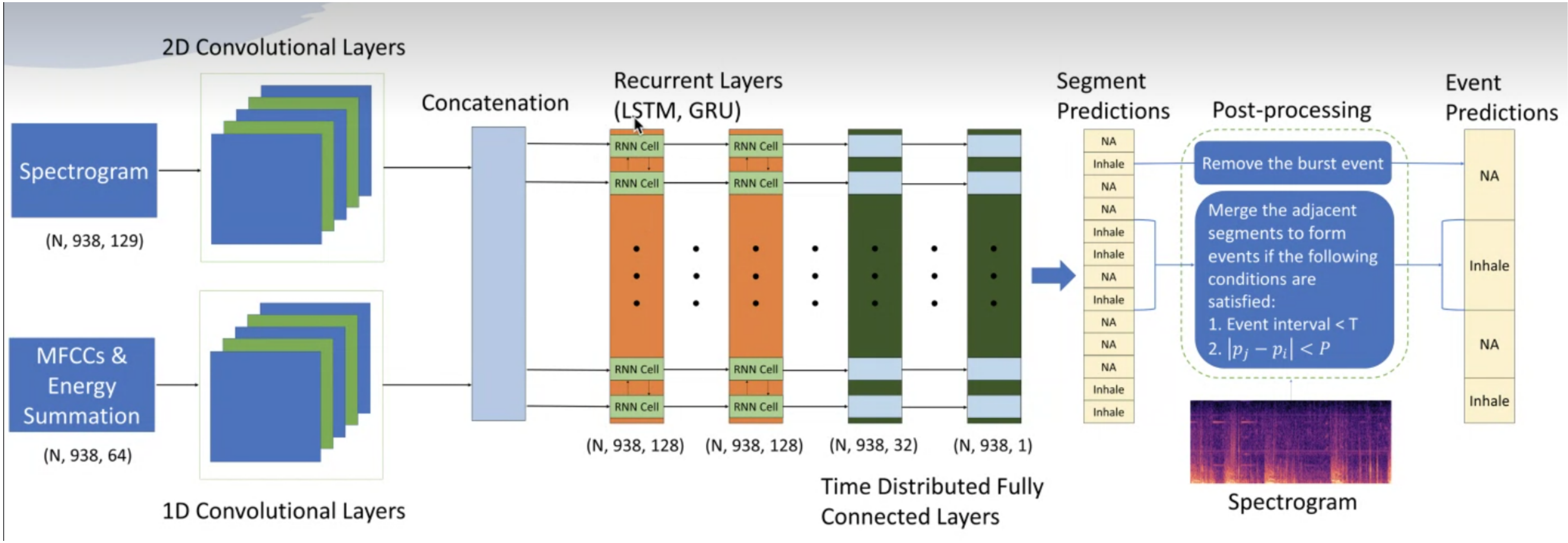
```
FILE *fp_out = fopen("S2.array", "wb");
fwrite(BUF_power_to_db, SIZE_BUF_power_to_db*sizeof(double), 1, fp_out);
fclose(fp_out);
```

Step 5-1: Airmod read processed data from disk

```
S2 = np.fromfile('S2.array').reshape(128,938)
if np.array_equal(S , S2):
     print("Host-Info: Test Successful")
else:
     print("Host-Error: Test Failed")
```

- S is golden result obtained by original Python code, S2 is result obtained by kernel
- Software emulation success

Step 5-2: Airmod feed processed data to machine learning model



Merge the adjacent segments to form events if the following conditions are satisfied:

1. Event interval < T
2. $|p_j - p_i| < P$

Result

- Software emulation success
- Hardware emulation have wrong value

```
S =
[[-57.99985373 -62.38820195 -71.39640441 ... -71.39640441 -46.10319584
  -29.98084382]
 [-61.70898922 -66.26478886 -60.9017189  ... -66.33469663 -45.86942947
  -29.18327811]
 [-41.99556866 -42.85118638 -44.46747413 ... -47.67435414 -42.63794043
  -27.24027219]
 ...
 [-62.53210127 -68.70541344 -71.39640441 ... -71.39640441 -71.39640441
  -68.70012327]
 [-62.88395057 -68.66971149 -71.39640441 ... -71.39640441 -71.39640441
  -67.13671548]
 [-63.088656   -69.1723574  -71.39640441 ... -71.39640441 -71.39640441
  -67.33849479]]
```

```
S2 =
[[1.58494657e-06 5.77005303e-07 1.63309445e-09 ... 2.37664272e-08
  2.45290324e-05 1.00442062e-03]
 [6.74685035e-07 2.36331229e-07 8.12508869e-07 ... 2.32557493e-07
  2.58855295e-05 1.20690251e-03]
 [6.31601473e-05 5.18658335e-05 3.57480690e-05 ... 1.70830175e-05
  5.44760936e-05 1.88787303e-03]
 ...
 [5.58200052e-07 1.34728246e-07 6.29370041e-10 ... 5.67968658e-09
  4.10374181e-09 1.34892459e-07]
 [5.14760178e-07 1.35840369e-07 1.37220771e-09 ... 9.52569933e-09
  1.23505759e-08 1.93342999e-07]
 [4.91059820e-07 1.20994118e-07 2.61981242e-09 ... 3.46911983e-09
  7.87195200e-09 1.84565499e-07]]
```

Other system integration method

- Because write data to disk have large I/O time, we want to access data of Python in memory directly in OpenCL host code

a. PyOpenCL

b. Python XRT binding

c. Compiled host code into shared object, then load in Python code

a. System integration by PyOpenCL

- We use PyOpenCL to integrate origin Python code with HLS C++ code
- When running software emulation Python host program, it can’t find emconfig.json, thus emulate using default device (not U50)
  - In Lab3, host program is compiled from C++, and it will find emconfig.json in the same directory of host program
  - In this project, host program is Python code with PyOpenCL, we don’t know which directory to place emconfig.json

```
[MAKEFILE_INFO] check input host=opencl target=sw_emu precision=7e-9
make[1]: ‘emconfig.json’ is up to date.
[Python Info] Use opencl host
CRITICAL WARNING: [SW-EM 09-0] Unable to find emconfig.json.
Using default device “Xilinx:pcie-hw-em:7v3:1.0”
Traceback (most recent call last):
  File “check.py”, line 66, in <module>
    host = Host_Opencl(target=args.target)
AssertionError
```

b. System integration by Python XRT binding

- We call XRT library directly in Python code
- But it still runs into error that we cannot solve

```
XRT build version: 2.7.766
[Thu Jan 21 11:49:21 2021 GMT]
Host: fpgamaster
EXE: /usr/bin/python3.6
[XRT] ERROR: See dmesg log for details. err=-22
Traceback (most recent call last):
  File “check.py”, line 71, in <module>
    host = Host_XRT(target = args.target)
OSError: [Errno 22] Invalid argument
```

c. System integration by compiled host code to shared object

- We re-write host program in C++ and compile it into shared object which would be called as shared library in Python code

```
[MAKEFILE_INFO] generate libhost.so
make[1]: ‘emconfig.json’ is up to date.
[Python Info] Use c_library host
Terminate called after throwing an instance of ‘std::runtime_error’
  what(): singleton ctor error
Aborted (core dumped)
make: *** Error 134
```