

Communication Training System

Design Document

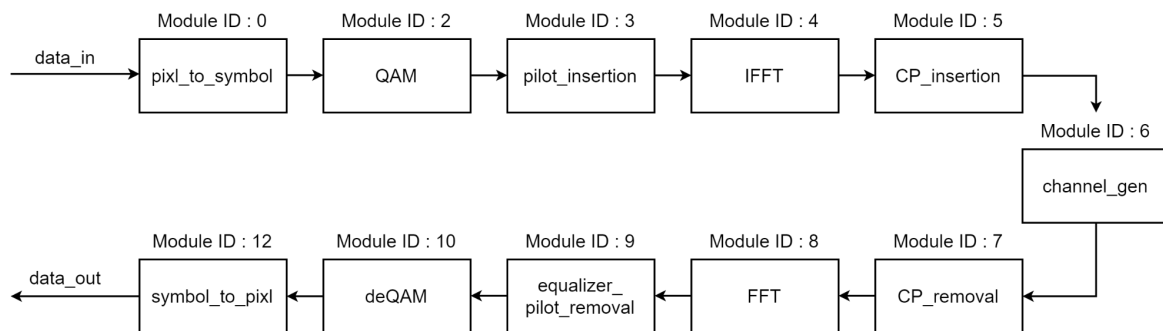
Table of Contents

- **System Overview**
 - **System Block Diagram**
 - **Feature List**
- **Module IP Documentation**
 - **Parameter Description**
 - **Function Description**
 - **Hardware Utilization for Each Kernel**
- **Host Software Overview**
 - **Host Code Description**

System Overview

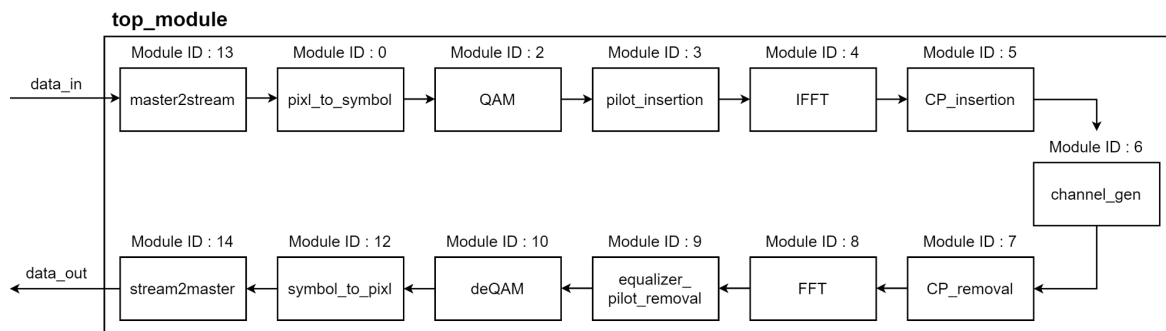
> System Block Diagram

◦ Composable Pipeline



- Input : hls::stream<ap_axiu<ap_uint<64>,0,0,0>> data_in
- Output : hls::stream<ap_axiu<ap_uint<64>,0,0,0>> data_out
- Block Level Control Protocols : ap_ctrl_none

◦ Direct Connection



- Input : ap_uint<64>* data_in
- Output : ap_uint<64>* data_out
- HLS pragma : dataflow (in top_module)
- Block Level Control Protocols : ap_ctrl_none (stream interface)

Module IP Documentation

> Data Structure Description

◦ Parameter

| module_id | parameter_id | parameter |
|-----------|--------------|-----------|
| 16bits | 16bits | 32bits |

◦ Data

| | |
|---------------------|---------------------|
| real part 32bits | imag part 32bits |
|---------------------|---------------------|

◦ Parameter stream

- Kernel

do{

```

para_in = data_in.read();
module_id = para_in.range(63, para_id_bit+para_val_bit);
if(module_id == module_num){
    para_id = para_in.range(para_id_bit+para_val_bit-1, para_val_bit);
    para_val = para_in.range(para_val_bit-1, 0);
    if(para_id == 0){
        ____ = para_val; //將讀出來的數值給對應參數名稱
    }
    else if(para_id == 1){
        ____ = para_val; //將讀出來的數值給對應參數名稱
    }
    :
    :
}
else{
    data_out.write(para_in);
}
}while( module_id != threshold);

```

- Testbench

```

module_id = __; //對應module的ID
para_id = __; //該參數在module內的排序
para_val = __; //參數數值
para_in = (module_id, para_id, para_val); //擺放成特定形式

```

1. for composable pipeline

```
data_in.read(para_in); //寫進input stream
```

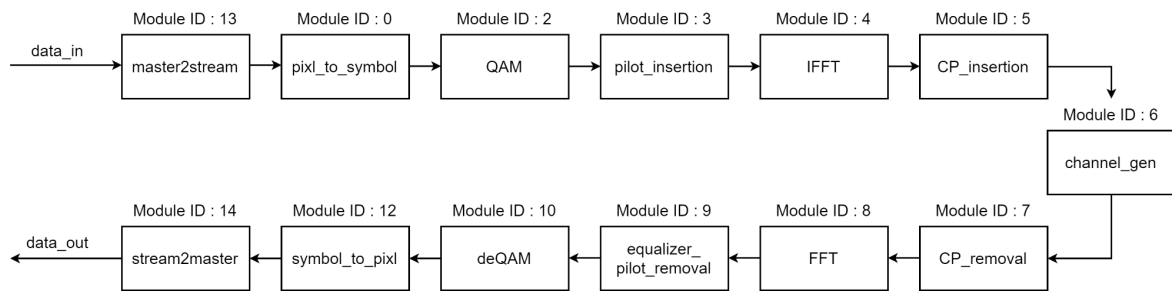
2. for direct connection

```
data_in[cnt] = para_in; //寫進input buffer
```

> Parameter Description

- data_len : 輸入訊號的總長度
- total_data_len : 加上參數以及終止條件後的總資料長度
- qam_num : 調變所使用的QAM數
- sym_num : 一個pixel對應到的symbol數
- pilot_width : 兩個pilot中間的間隔
- CP_length : CP的長度
- TAPS_NUM : 多路徑通道的TAP數
- SNR : 訊號與雜訊的能量比, 用dB表示

> Function Description



。pixl2sym

- Module ID : 0

- Feature : 將8位元的input pixel轉換成特定位元數的symbol, symbol的位元數會取決於所使用的調變方法 (QAM)。

- Parameters :

ID = 0, parameter = data_len

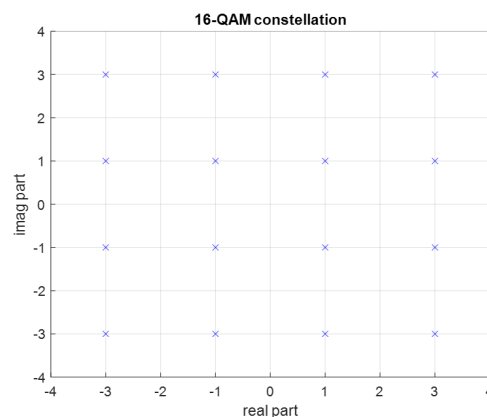
ID = 1, parameter = qam_num

ID = 2, parameter = sym_num

。QAM

- Module ID : 2

- Feature : 在兩個正交載波上進行震幅調變。



- Parameters :

ID = 0, parameter = data_len

ID = 1, parameter = qam_num

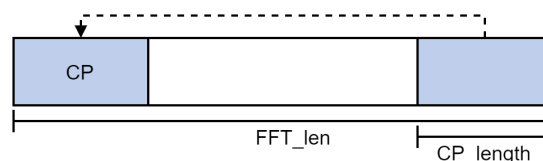
ID = 2, parameter = sym_num

ID = 3, parameter = pilot_width

。pilot_insertion

- Module ID : 3

- Feature : 在相隔pilot_width-1的位置插入pilot (real=1, imag=0)。



- Parameters :

ID = 0, parameter = data_len

ID = 1, parameter = sym_num

ID = 2, parameter = pilot_width

ID = 3, parameter = CP_length

。 IFFT

- Module ID : 4

- Feature : 將資料從頻域轉換至時域 (By Xilinx DSP Library)。

- Parameters :

ID = 0, parameter = data_len

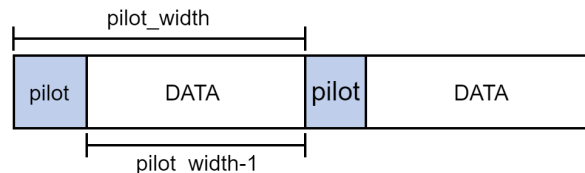
ID = 1, parameter = sym_num

ID = 2, parameter = pilot_width

。 CP_insertion

- Module ID : 5

- Feature : 用於在每個symbol之前插入長度為CP_length的guard band, CP會取自於該symbol的最尾端CP_length個data。



- Parameters :

ID = 0, parameter = data_len

ID = 1, parameter = sym_num

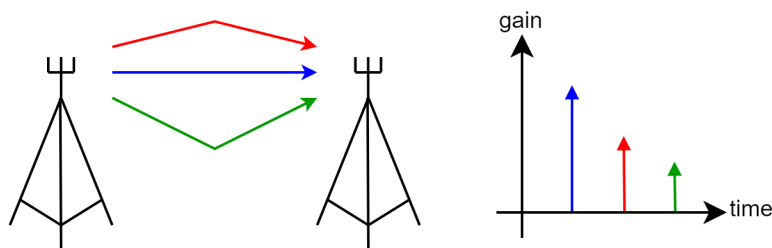
ID = 2, parameter = pilot_width

ID = 3, parameter = CP_length

。 channel_gen

- Module ID : 6

- Feature : 用於模擬無線通訊系統中的多路徑效應。



- Parameters :

ID = 0, parameter = data_len

ID = 1, parameter = sym_num

ID = 2, parameter = pilot_width

ID = 3, parameter = CP_length

ID = 4, parameter = TAPS_NUM

ID = 5, parameter = SNR

- Channel Fading Type : unit gain channel, multipath channel

- Channel Gain for Multi-path Channel :

2-tap = [0.85, 0.5267]

3-tap = [0.8, 0.5, 0.3317]

6-tap = [0.7943, 0.3981, 0.3162, 0.2512, 0.1778, 0.1259]

9-tap = [0.5, 0.31, 0.19, 0.06, 0.03, 0.02, 0.008, 0.004, 0.002]

◦ **CP_removal**

- Module ID : 6

- Feature : 移除CP。

- Parameters :

ID = 0, parameter = data_len

ID = 1, parameter = sym_num

ID = 2, parameter = pilot_width

ID = 3, parameter = CP_length

◦ **FFT**

- Module ID : 8

- Feature : 將資料從時域轉換至頻域 (By Xilinx DSP Library)。

- Parameters :

ID = 0, parameter = data_len

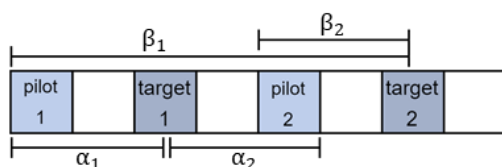
ID = 1, parameter = sym_num

ID = 2, parameter = pilot_width

◦ **equalizer_pilot_removal**

- Module ID : 9

- Feature : 利用已知pilot並使用內插法以及外插法來估算通道的效應，移除通道效應並將pilot部分從資料中移除。



• Interpolation

$$\text{Channel gain of } target_1 = \frac{\alpha_1 \times pilot_1 + \alpha_2 \times pilot_2}{\alpha_1 + \alpha_2}$$

• Extrapolation

$$\text{Channel gain of } target_2 = \frac{\beta_1 \times pilot_1 + \beta_2 \times pilot_2}{\beta_1 + \beta_2}$$

- Parameters :

ID = 0, parameter = data_len

ID = 1, parameter = sym_num

ID = 2, parameter = pilot_width

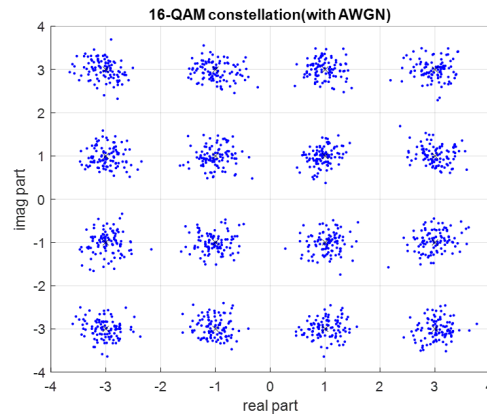
ID = 3, parameter = CP_length

ID = 4, parameter = TAPS_NUM

◦ **deQAM**

- Module ID : 10

- Feature : 透過判斷震幅大小將調變過的訊號還原。



- parameters :

ID = 0, parameter = data_len

ID = 1, parameter = qam_num

ID = 2, parameter = sym_num

ID = 3, parameter = pilot_width

◦ **sym2pixl**

- Module ID : 12

- Feature : 將deQAM完成的symbol合併成pixel。

- Parameters :

ID = 0, parameter = data_len

ID = 1, parameter = qam_num

ID = 2, parameter = sym_num

◦ **master2stream**

- Module ID : 13

- Feature : 將從host讀到的master資料讀進stream, 以便後續module使用。

- Parameters :

ID = 0, parameter = total_data_len

◦ **stream2master**

- Module ID : 14

- Feature : 將完成計算的stream資料轉成master的形式, 為後續傳回host端進行準備。

- Parameters :

ID = 0, parameter = data_len

> Utilization for Each Kernel

◦ Device : KV260

◦ Clock period : 20ns

◦ Composable Pipeline (stream interface)

| Name | BRAM | DSP | FF | LUT |
|-----------------|------|-----|-------|-------|
| pixl2sym | 0 | 0 | 18241 | 13733 |
| QAM | 0 | 3 | 405 | 621 |
| pilot_insertion | 0 | 11 | 3544 | 3278 |

| | | | | |
|---------------|-----|-----|-------|-------|
| IFFT | 16 | 23 | 11367 | 13146 |
| CP_insertion | 2 | 3 | 1266 | 1257 |
| channel_gen | 11 | 102 | 6025 | 16118 |
| CP_removal | 0 | 6 | 1321 | 1323 |
| FFT | 16 | 23 | 11367 | 12998 |
| pilot_removal | 0 | 35 | 41995 | 27717 |
| deQAM | 0 | 3 | 791 | 1169 |
| sym2pixl | 0 | 3 | 284 | 402 |
| total | 53 | 212 | 96606 | 91762 |
| total (%) | 18% | 17% | 41% | 78% |

。 Direct Connection (master interface)

| Name | BRAM | DSP | FF | LUT |
|-----------------|-------------|------------|-----------|------------|
| master2stream | 0 | 0 | 335 | 587 |
| pixl2sym | 0 | 3 | 17987 | 13685 |
| QAM | 0 | 3 | 336 | 475 |
| pilot_insertion | 0 | 11 | 3524 | 3192 |
| IFFT | 16 | 23 | 11253 | 13174 |
| CP_insertion | 2 | 3 | 1230 | 1139 |
| channel_gen | 11 | 101 | 6218 | 16399 |
| CP_removal | 0 | 6 | 1237 | 1237 |
| FFT | 16 | 23 | 11253 | 13026 |
| pilot_removal | 0 | 36 | 40147 | 26559 |
| deQAM | 0 | 3 | 766 | 1119 |
| sym2pixl | 0 | 3 | 208 | 322 |
| entry_proc | 0 | 0 | 66 | 20 |
| stream2master | 0 | 0 | 367 | 518 |
| total | 53 | 215 | 96958 | 93398 |

| | | | | |
|-----------|-----|-----|-----|-----|
| total (%) | 18% | 17% | 41% | 79% |
|-----------|-----|-----|-----|-----|

Host Code Description

- Import IP

填入和vivado對應的ip名稱以及.bit檔名稱, .bit的檔名需要和.hwh一致。

```
ol = Overlay("if_TAP_1.bit")
ipcomm = ol.top_module_0
```

- Read data

在jupyter notebook直接將圖片轉換成pixel資料。

```
RGB_val = 3
print("#-----ORIGIANL PIC-----#")
cat_pic = Image.open("cat_punch.jpg")
plt.imshow(cat_pic)
plt.show()
cat_pic = cat_pic.rotate(270)
cat_pic = cat_pic.transpose(Image.FLIP_LEFT_RIGHT)
pic_array = allocate(shape=((cat_pic.size[0]*cat_pic.size[1]*3),), dtype=np.uint8)
for inx1 in range(cat_pic.size[0]):
    for inx2 in range(cat_pic.size[1]):
        pix = cat_pic.getpixel((inx1,inx2))
        pic_array[(inx1)*(cat_pic.size[1])*RGB_val+(inx2)*RGB_val+0] = pix[0]
        pic_array[(inx1)*(cat_pic.size[1])*RGB_val+(inx2)*RGB_val+1] = pix[1]
        pic_array[(inx1)*(cat_pic.size[1])*RGB_val+(inx2)*RGB_val+2] = pix[2]
print("#-----READ DATA OK-----#")
```

- Parameter setting

設定系統裡面會使用到的參數。

```
qam_num = 16
sym_num = 2
pilot_width = 4
CP_length = 16
TAPS_NUM = 1
SNR = [15]
FFT_len = 64
```

- Buffer allocation

由於輸入的資料量需要是 $(FFT_LEN - (FFT_LEN / pilot_width)) / sym_num$ 的倍數, 為了確保kernel執行時不會出現錯誤, 要事先將不足的數量計算出來, 在後續設定input buffer時將不足的部分補0。

```
numSamples = cat_pic.size[0]*cat_pic.size[1]*3
input_compensate = (int)((((FFT_len-(FFT_len/pilot_width))/sym_num)-
                        (numSamples%((FFT_len-(FFT_len/pilot_width))/sym_num))))
data_len = numSamples + input_compensate
total_input_num = data_len + parameter_sum + 1
in_num_sample = parameter_sum + 1 + data_len
out_num_sample = data_len

input_buffer = allocate(shape=(in_num_sample,), dtype=np.uint64)
output_buffer = allocate(shape=(out_num_sample,), dtype=np.uint64)
```

- Input buffer setting

根據各個kernel的需求設定對應的參數，並將參數寫入input buffer。

```
parameter_id = np.array([[data_len,qam_num,sym_num], # module_id = 0 pixl2sym
                          [data_len,qam_num,sym_num,pilot_width], # module_id = 2 QAM
                          [data_len,sym_num,pilot_width,CP_length], # module_id = 3 pilot insert
                          [data_len,sym_num,pilot_width], # module_id = 4 IFFT
                          [data_len,sym_num,pilot_width,CP_length], # module_id = 5 CP insert
                          [data_len,sym_num,pilot_width,CP_length,TAPS_NUM,SNR], # module_id = 6 channel
                          [data_len,sym_num,pilot_width,CP_length], # module_id = 7 CP remove
                          [data_len,sym_num,pilot_width], # module_id = 8 FFT
                          [data_len,sym_num,pilot_width,CP_length, TAPS_NUM], # module_id = 9 pilot remove
                          [data_len,qam_num,sym_num,pilot_width], # module_id = 10 deQAM
                          [data_len,qam_num,sym_num], # module_id = 12 sym2pixl
                          [total_input_num], # module_id = 13 mas2str
                          [data_len]],dtype=object) # module_id = 14 str2mas

#-----parameter set-----#
temp = allocate(shape=(1,), dtype=np.uint64)
cnt_para_in = 0
for i in range(len_module_id):
    temp[0] = module_id[i] * pow(2,16)
    for j in range(len_parameter_id[i]):
        input_buffer[cnt_para_in+j] = temp[0]
        input_buffer[cnt_para_in+j] = input_buffer[cnt_para_in+j] + j
        input_buffer[cnt_para_in+j] = input_buffer[cnt_para_in+j] * pow(2,32)
        input_buffer[cnt_para_in+j] = input_buffer[cnt_para_in+j] + parameter_id[i][j]
    cnt_para_in = cnt_para_in+len_parameter_id[i]
input_buffer[parameter_sum] = (pow(2,16)-1)*pow(2,48)
```

將圖片資料寫在parameter_stream的後面，並將不足的部分補0。

```
#-----data set-----#
for k in range(data_len):
    if (k<numSamples):
        input_buffer[parameter_sum+k+1] = pic_array[k]
    else:
        input_buffer[parameter_sum+k+1] = 0
```

- Run kernel

將input_buffer裡面的data寫入pynq裡面，等待kernel執行完成之後讀取ouput_buffer裡面的數值，並對kernel運行時間進行計時。

◦ Composable pipeline

```
print("#-----KERNEL START-----#")
dma_send.transfer(input_buffer)
dma_recv.transfer(output_buffer)
dma_send.wait()
dma_recv.wait()
print("#-----KERNEL END-----#")
```

◦ Direct connection

```
print("#-----KERNEL START-----#")
timeKernelStart = time()
ipcomm.write(0x10, input_buffer.device_address)
ipcomm.write(0x1C, output_buffer.device_address)
ipcomm.write(0x00, 0x01)
while (ipcomm.read(0x00) & 0x4) == 0x0:
    continue
timeKernelEnd = time()
print("Kernel execution time: " + str(timeKernelEnd - timeKernelStart) + " s")
print("#-----KERNEL END-----#")
```

- Output analysis

最後要計算錯誤率，在通訊領域裡面通常是以bit error rate(BER)作為基準，因此將symbol中的每個bit都提出來比較，並且將ouput data還原成圖片以便觀察。

```
ber = 0
pixl_length = 8
for k in range(numSamples):
    err = '{:08b}'.format(output_buffer[k]^(input_buffer[k+parameter_sum+1])).count('1')
    ber = ber + err
    out_pic[k] = output_buffer[k]
BER[SNR_index] = ber/numSamples/pixl_length*100

out_pic_matrix = out_pic.reshape(cat_pic.size[0],cat_pic.size[1],3)
plt.imshow(out_pic_matrix)
plt.show()
```