Communication Training System

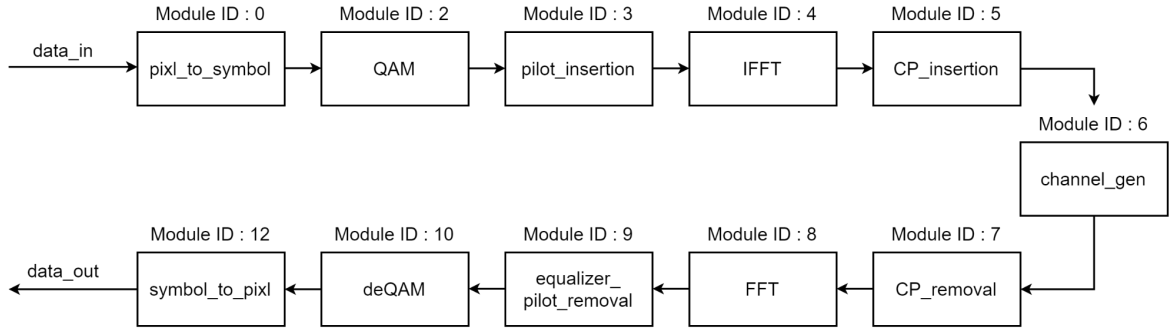
Design Document

Table of Contents

System Overview

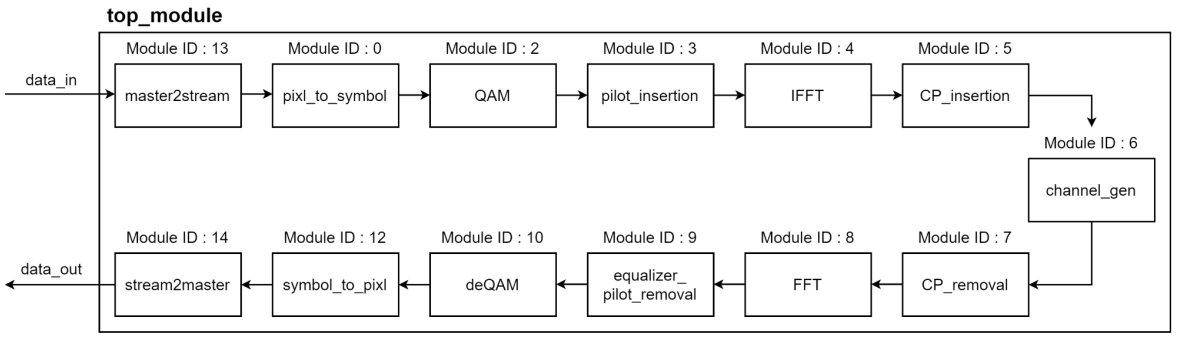
> System Block Diagram

◦ 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	imag part
32bits	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);
```

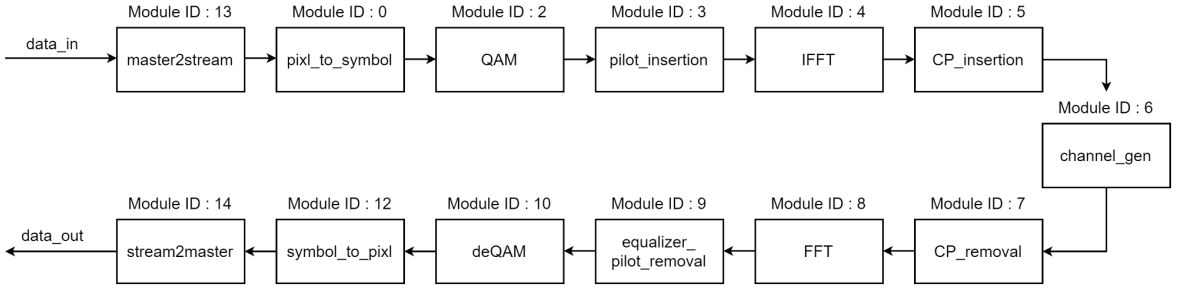
- Testbench

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

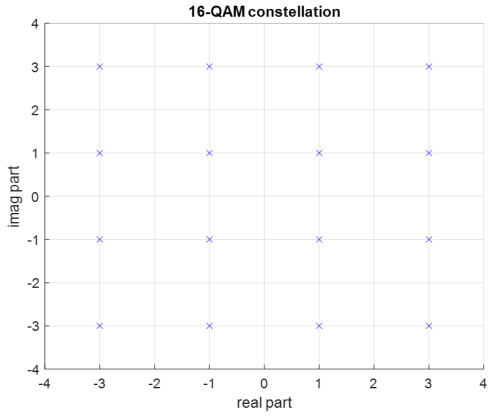
ID = 1, parameter = qam_num

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

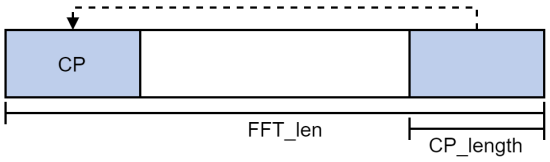
ID = 2, parameter = sym_num

ID = 3, parameter = pilot_width

。**pilot_insertion**

- Module ID : 3

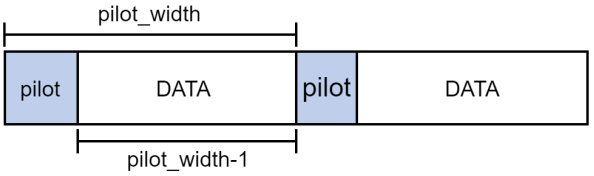
- Feature : 在相隔pilot_width-1的位置插入pilot (real=1, imag=0)。



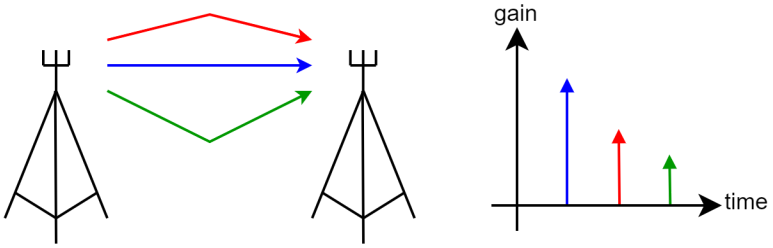
- Parameters :

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

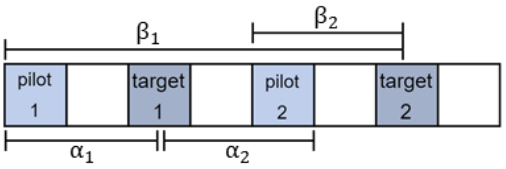
。**FFT**

- 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

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

- Extrapolation

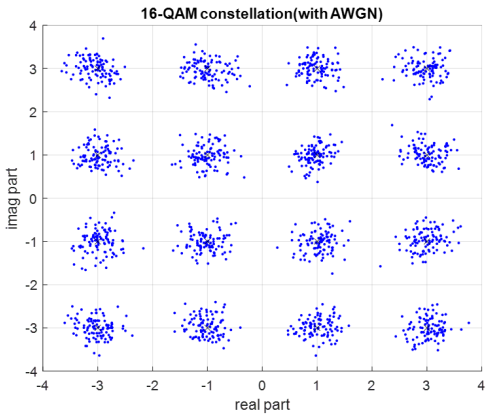
Channel gain of $target_2 = \frac{\beta_1 \times pilot_1 + \beta_2 \times pilot_2}{\beta_1 + \beta_2}$

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

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

。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%

> Latency for each kernel

。Composable Pipeline (stream interface)

Name	Avg Latency	Max Latency	Min Latency
master2stream			
pixl2sym			
QAM			
pilot_insertion			
IFFT			
CP_insertion			
channel_gen			
CP_removal			
FFT			
pilot_removal			
deQAM			
sym2pixl			
stream2master			
total			

。Direct Connection (master interface)

Name	Avg Latency	Max Latency	Min Latency
master2stream	14968	14968	14968
pixl2sym	26172	26172	26172
QAM	26447	26447	26447
pilot_insertion	26634	26634	26634
IFFT	36506	36506	36506
CP_insertion	42771	42771	42771
channel_gen	42960	42960	42960

CP_removal	42958	42958	42958
FFT	46491	46491	46491
pilot_removal	47572	47572	47572
deQAM	47609	47609	47609
sym2pixl	47609	47609	47609
stream2master	47636	47636	47636
total	47637	47637	47637

Host Code Description

。Composable Pipeline (stream interface)

。Direct Connection (master interface)

- 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

- Input data compensation

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，將不足的資料數補0。

- Run kernel

將input_buffer裡面的data寫入pynq裡面，並等待kernel執行完成之後讀取ouput_buffer裡面的數值。

```
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。