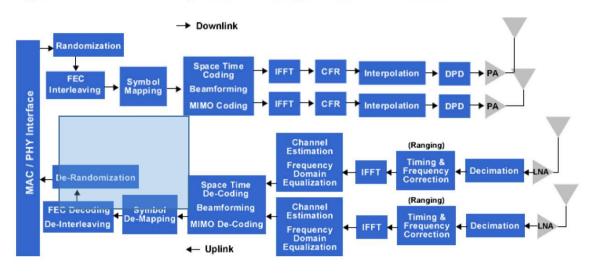
Project Specifications¹

WiMax PHY—Channel Coding

1. Introduction

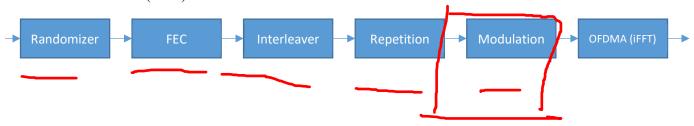
In this project, you are required to implement part of the PHY layer of a WiMax system; namely "Channel Coding" (QPSK only) as described in section 8.4.9 in WiMax Standard (IEEE Std 802.16-2007). This part of the standard is explained in this document with some numeric examples. WiMax PHY including "Channel Coding" has different parameters that are set by the MAC layer. There are five different blocks within the "Chanel Coding", each has different parameters that might require several implementations. In this project you are only required to implement one implementation per block.

Figure 3: Overview of PHY Layer functions in a typical WiMAX base station



2. Channel Coding²

- Channel coding procedures include:
 - 1. Randomization (see 8.4.9.1).
 - 2. FEC encoding (see 8.4.9.2).
 - 3. Bit interleaving (see 8.4.9.3).
 - 4. Repetition (see 8.4.9.5), only applied to QPSK modulation.
 - 5. Modulation (see 8.4.9.4).
 - 6. Orthogonal Frequency Division Multiple Access (OFDMA); Inverse Fast Fourier Transform (iFFT)



¹ Revision 2023 11 16

² See section (8.4.9)

3. Randomizer³

A. Initializing Randomization

- The randomization is initialized on each FEC block.
- If the amount of data to transmit does not fit exactly the amount of data allocated, padding of 0xFF ("1" only) shall be added to the end of the transmission block, up to the amount of data allocated.
 - Here, the amount of data allocated means the amount of data that corresponds to the amount of $\lfloor N_s / R \rfloor$ slots, where N_s is the number of the slots allocated for the data burst and R is the repetition factor used.

B. RTL Requirements

- The PRBS generator shall be $1 + X^{14} + X^{15}$ as shown in Figure 253.
- Each data byte to be transmitted shall enter sequentially into the randomizer, MSB first.
- The seed value shall be used to calculate the randomization bits, which are combined in an XOR operation with the serialized bit stream of each FEC block.

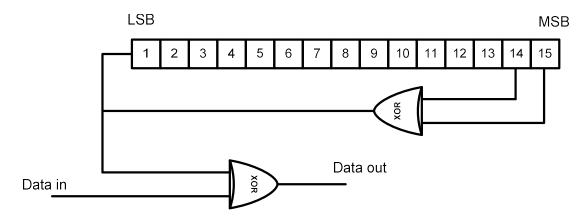


Figure 253—PRBS generator for data randomization

- The randomizer is initialized with the vector:
 - o [LSB] 0 1 1 0 1 1 1 0 0 0 1 0 1 0 1 [MSB].
- The bit issued from the randomizer shall be applied to the encoder.
- The randomizer block should be implemented using the block diagram below. The block has a 50 MHz clock and an asynchronous reset. The input "rand_in_ready" is asserted when there is a valid input data "rand_in". The output "rand_out_valid" is asserted when there a valid data

C. Testbench Gold Data:

- Number of symbols Ns = 2 symbols
- Symbol size = 48 bits
- Block size = Ns * 48 = 96 bits

³ Section 8.4.9.1 in the standard

- Input Data (Hex):
 - o AC BC D2 11 4D AE 15 77 C6 DB F4 C9
- Randomized Data (Hex):
 - o 55 8A C4 A5 3A 17 24 E1 63 AC 2B F9

802.16 e

Ns = 2 slots 96 bits Init

Initialized vector 011 0111 0001 0101

No	Shi	ft Re	gister													XOR	Dat	a in	Data out	a
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	#1	0x	0b	0b	0x
1	0	1	1	0	1	1	1	0	0	0	1	0	1	0	1	1		1	0	
2	1	0	1	1	0	1	1	1	0	0	0	1	0	1	0	1	١,	0	1	5
3	1	1	0	1	1	0	1	1	1	0	0	0	1	0	1	1	A	1	0	3
4	1	1	1	0	1	1	0	1	1	1	0	0	0	1	0	1		0	1	
5	1	1	1	1	0	1	1	0	1	1	1	0	0	0	1	1		1	0	
6	1	1	1	1	1	0	1	1	0	1	1	1	0	0	0	0	$ _{\mathcal{C}}$	1	1	5
7	0	1	1	1	1	1	0	1	1	0	1	1	1	0	0	0		0	0	3
8	0	0	1	1	1	1	1	0	1	1	0	1	1	1	0	1		0	1	
9	1	0	0	1	1	1	1	1	0	1	1	0	1	1	1	0		1	1	
10	0	1	0	0	1	1	1	1	1	0	1	1	0	1	1	0	В	0	0	8
11	0	0	1	0	0	1	1	1	1	1	0	1	1	0	1	1	D	1	0	0
12	1	0	0	1	0	0	1	1	1	1	1	0	1	1	0	1		1	0	
13	1	1	0	0	1	0	0	1	1	1	1	1	0	1	1	0		1	1	
14	0	1	1	0	0	1	0	0	1	1	1	1	1	0	1	1	$ _{\mathcal{C}}$	1	0	A
15	1	0	1	1	0	0	1	0	0	1	1	1	1	1	0	1		0	1	A
16	1	1	0	1	1	0	0	1	0	0	1	1	1	1	1	0		0	0	
17	0	1	1	0	1	1	0	0	1	0	0	1	1	1	1	0		1	1	
18	0	0	1	1	0	1	1	0	0	1	0	0	1	1	1	0	D	1	1	C
19	0	0	0	1	1	0	1	1	0	0	1	0	0	1	1	0	שן	0	0	
20	0	0	0	0	1	1	0	1	1	0	0	1	0	0	1	1		1	0	
21	1	0	0	0	0	1	1	0	1	1	0	0	1	0	0	0		0	0	
22	0	1	0	0	0	0	1	1	0	1	1	0	0	1	0	1	2	0	1	4
23	1	0	1	0	0	0	0	1	1	0	1	1	0	0	1	1	~	1	0	-
24	1	1	0	1	0	0	0	0	1	1	0	1	1	0	0	0		0	0	
25	0	1	1	0	1	0	0	0	0	1	1	0	1	1	0	1		0	1	
26	1	0	1	1	0	1	0	0	0	0	1	1	0	1	1	0	1	0	0	A
27	0	1	0	1	1	0	1	0	0	0	0	1	1	0	1	1	1	0	1	Α
28	1	0	1	0	1	1	0	1	0	0	0	0	1	1	0	1		1	0	
29	1	1	0	1	0	1	1	0	1	0	0	0	0	1	1	0		0	0	
30	0	1	1	0	1	0	1	1	0	1	0	0	0	0	1	1	1	0	1	5
31	1	0	1	1	0	1	0	1	1	0	1	0	0	0	0	0	1	0	0	
32	0	1	0	1	1	0	1	0	1	1	0	1	0	0	0	0		1	1	
33	0	0	1	0	1	1	0	1	0	1	1	0	1	0	0	0		0	0	
34	0	0	0	1	0	1	1	0	1	0	1	1	0	1	0	1	4	1	0	3
35	1	0	0	0	1	0	1	1	0	1	0	1	1	0	1	1	¯	0	1	
36	1	1	0	0	0	1	0	1	1	0	1	0	1	1	0	1		0	1	
37	1	1	1	0	0	0	1	0	1	1	0	1	0	1	1	0	D	1	1	Α

38	0	1	1	1	0	0	0	1	0	1	1	0	1	0	1	1		1	0	
39	1	0	1	1	1	0	0	0	1	0	1	1	0	1	0	1		0	1	
40	1	1	0	1	1	1	0	0	0	1	0	1	1	0	1	1	İ	1	0	
41	1	1	1	0	1	1	1	0	0	0	1	0	1	1	0	1	1	1	0	
42																1		1	1	
	1	1	1	1	0	1	1	1	0	0	0	1	0	1	1	0	Α	0	0	1
43	0	1	1	1	1	0	1	1	1	0	0	0	1	0	1	1		1	0	
44	1	0	1	1	1	1	0	1	1	1	0	0	0	1	0	1		0	1	
45	1	1	0	1	1	1	1	0	1	1	1	0	0	0	1	1		1	0	
46	1	1	1	0	1	1	1	1	0	1	1	1	0	0	0	0	_	1	1	_
47	0	1	1	1	0	1	1	1	1	0	1	1	1	0	0	0	E	1	1	7
48	0	0	1	1	1	0	1	1	1	1	0	1	1	1	0	1		0	1	
49	1	0	0	1	1	1	0	1	1	1	1	0	1	1	1	0	+	0	0	
1	1											0				i		1	i	
50	0	1	0	0	1	1	1	0	1	1	1	1	0	1	1	0	1	0	0	2
51	0	0	1	0	0	1	1	1	0	1	1	1	1	0	1	1	1	0	1	-
52	1	0	0	1	0	0	1	1	1	0	1	1	1	1	0	1		1	0	
53	1	1	0	0	1	0	0	1	1	1	0	1	1	1	1	0		0	0	
54	0	1	1	0	0	1	0	0	1	1	1	0	1	1	1	0	İ	1	1	
55	0	0	1	1	0	0	1	0	0	1	1	1	0	1	1	0	5	0	0	4
1	i							_								i	1	1	1	
56	0	0	0	1	1	0	0	1	0	0	1	1	1	0	1	1	 	1	0	
57	1	0	0	0	1	1	0	0	1	0	0	1	1	1	0	1		0	1	
58	1	1	0	0	0	1	1	0	0	1	0	0	1	1	1	0	7	1	1	E
59	0	1	1	0	0	0	1	1	0	0	1	0	0	1	1	0	′	1	1	
60	0	0	1	1	0	0	0	1	1	0	0	1	0	0	1	1	1	1	0	
61	1	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0		0	0	
62	0	1	0	0	1	1	0	0	0	1	1	0	0	1	0	1		1	0	
63	1 .																7		1	1
1	1	0	1	0	0	1	1	0	0	0	1	1	0	0	1	1		1	0	
64	1	1	0	1	0	0	1	1	0	0	0	1	1	0	0	0	+	1	1	
65	0	1	1	0	1	0	0	1	1	0	0	0	1	1	0	1		1	0	
66	1	0	1	1	0	1	0	0	1	1	0	0	0	1	1	0	C	1	1	6
67	0	1	0	1	1	0	1	0	0	1	1	0	0	0	1	1		0	1	
68	1	0	1	0	1	1	0	1	0	0	1	1	0	0	0	0	1	0	0	
69	0	1	0	1	0	1	1	0	1	0	0	1	1	0	0	0		0	0	
70	0	0	1	0	1	0	1	1	0	1	0	0	1	1	0	1	1	1	0	
i		_	_	_		4							_		1	_	6	1	1 4	3
71	l	0	0	1	0	I	0	1	l	0	l	0	0	l	1	0		1	I	
72	0	1	0	0	1	0	1	0	1	1	0	1	0	0	1	1	<u> </u>	0	1	
73	1	0	1	0	0	1	0	1	0	1	1	0	1	0	0	0	1	1	1	
74	0	1	0	1	0	0	1	0	1	0	1	1	0	1	0	1	D	1	0	,
75	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1	1	ע	0	1	A
76	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1	1	1	0	
77	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	+	1	1	
78	ł																	1	1	
	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	В	0	1	C
79	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1		1	0	
80	1	1	0	1	1	1	0	1	0	1	0	0	1	0	1	1	1	1	0	
81	1	1	1	0	1	1	1	0	1	0	1	0	0	1	0	1		1	0	
82	1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1		1	0	
83	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	F	1	1	2
84	0	1	1	1	1	1	0	1	1	1	0	1	0	1	0	1		1	0	
85	1	0	1	1	1	1	1	0	1	1	1	0	1	0	1	1	4	0	1	В
100	T	U	1	1	1	1	1	U	1	1	1	U	1	U	1	l 1	<u> </u>	J	l T	ט

86	1	1	0	1	1	1	1	1	0	1	1	1	0	1	0	1		1	0	
87	1	1	1	0	1	1	1	1	1	0	1	1	1	0	1	1		0	1	
88	1	1	1	1	0	1	1	1	1	1	0	1	1	1	0	1		0	1	
89	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	0		1	1	
90	0	1	1	1	1	1	(1	1	1	1	1	0	1	1	0	$ _{\mathcal{C}}$	1	1	$\mid_{\mathrm{F}}\mid$
91	0	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1		0	1	$\mid \Gamma \mid \mid$
92	1	0	0	1	1	1	1	1	0	1	1	1	1	1	0	1		0	1	
93	1	1	0	0	1	1	1	1	1	0	1	1	1	1	1	0		1	1	
94	0	1	1	0	0	1	1	1	1	1	0	1	1	1	1	0		0	0	
95	0	0	1	1	0	0	1	1	1	1	1	0	1	1	1	0	9	0	0	9
96	0	0	0	1	1	0	0	1	1	1	1	1	0	1	1	0		1	1	

•

4. FEC Encoder⁴

A. Tail-Biting Convolutional Coding

- The coding method used as the mandatory scheme will be the tail-biting convolutional encoding specified in 8.4.9.2.1.
- The encoding block size shall depend on the number of slot allocated and the modulation specified for the current transmission.
- Concatenation of a number of slots shall be performed in order to make larger blocks of coding where it is possible, with the limitation of not exceeding the largest supported block size for the applied modulation and coding.
- Table 318 specifies the concatenation of slots for different allocations and modulations.
- The parameters in Table 317 and Table 318 shall apply to the CC encoding scheme (see 8.4.9.2.1).

B. Definitions

- For any modulation and FEC rate, given an allocation of *n* slots, the following parameters are defined:
 - o j: parameter dependent on the modulation and FEC rate
 - o *n*: floor(number of allocated slots * STC rate/(repetition factor * number of STC layers))

$$k: \left| \frac{n}{j} \right|$$

- o m: n modulo j
- Table 317 shows the rules used for slot concatenation.

Table 317—Slots concatenation rule

Number of slots	Slots concatenated
$n \leq j$	1 block of <i>n</i> slots
n > j	If $(n \mod j = 0)$ $k \text{ blocks of } j \text{ slots}$ else $(k-1) \text{ blocks of } j \text{ slots}$ $1 \text{ block of } \left[\frac{m+j}{2}\right] \text{ slots}$ $1 \text{ block of } \left[\frac{m+j}{2}\right] \text{ slots}$

⁴ 8.4.9.2

Table 318—Encoding slot concatenation for different allocations and modulations

different differentials differ	modulations
Modulation and rate	j
QPSK 1/2	j=6
QPSK 3/4	j=4
16-QAM 1/2	j=3
16-QAM 3/4	j=2
64-QAM 1/2	j=2
64-QAM 2/3	j = 1
64-QAM 3/4	j = 1

Example 2a:

• From Example 1, we have QPSK with rate ½.

$$j = 6,$$
 $n = \left\lfloor \frac{N_s}{R} \right\rfloor = \left\lfloor \frac{2}{1} \right\rfloor = 2$ \Rightarrow $n \le j$

o 1 block of 2 slots:

• A block of $48 \times 2 = 96$ bits (12 bytes).

Example 2b:

• If we have $N_s = 12$, R = 1, and we use QPSK.

$$j = 6, n = \left\lfloor \frac{N_s}{R} \right\rfloor = \left\lfloor \frac{12}{1} \right\rfloor = 12 \qquad \Rightarrow \qquad n > j, k = \left\lfloor \frac{n}{j} \right\rfloor = \left\lfloor \frac{12}{6} \right\rfloor = 2, m = 0$$

o 2 block of 6 slots:

• 2 blocks of $48 \times 6 = 288$ bits (36 bytes).

C. Convolutional Coding⁵

• Each FEC block is encoded by the binary convolutional encoder, which shall have native rate of 1/2, a constraint length equal to K = 7, and shall use the following generator polynomials codes to derive its two code bits:

$$G_1 = 171_{OCT};$$
 For X
 $G_2 = 133_{OCT};$ For Y (125)

• The generator is depicted in Figure 255.

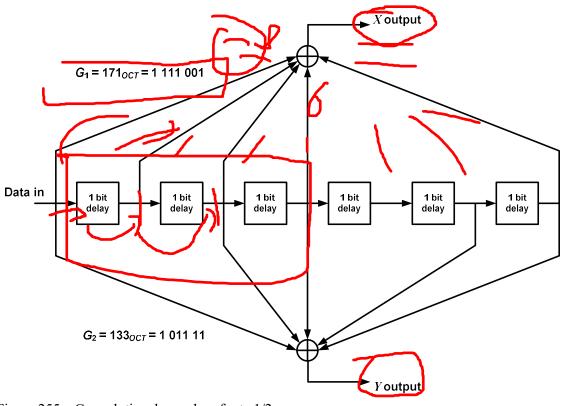


Figure 255—Convolutional encoder of rate 1/2

Puncturing

- The puncturing patterns and serialization order that shall be used to realize different code rates are defined in Table 319.
- In the table, "1" means a transmitted bit and "0" denotes a removed bit, whereas X and Y are in reference to Figure 255.

^{5 8.4.9.2.1}

Table 319—The convolutional code with

puncturing configuration

	- 0		
		Code Ra	tes
Rate	1/2	2/3	3/4
d_{free}	10	6	5
X	1	10	101
Y	1	11	110
XY	X_1Y_1	$X_1Y_1Y_2$	$X_1Y_1Y_2X_3$

Example 3:

• The figure below shows a puncturing encoder of rate 2/3.

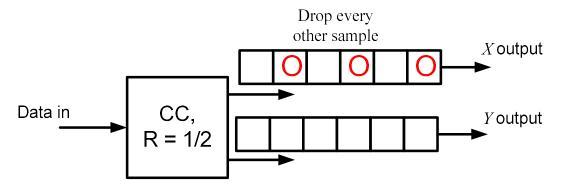


Figure 255b—Convolutional encoder of rate 2/3. The blocks inside are same as in Figure 255.

Tail-Biting Convolutional Coding

• Each FEC block is encoded by a tail-biting convolutional encoder, which is achieved by initializing the encoders' memory with the last data bits of the FEC block being encoded (the packet data bits numbered $b_{n-5}...b_n$).

Example:

- For example, assume that the last data bits in an FEC block are ... 0100 0110.
 - o Then the shift register is initialized as shown in Figure 255c.

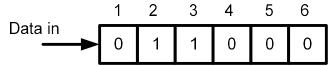


Figure 255c—Initialization example.

- The idea behind this is to make it as if the data are wrapped around itself.
- Table 320 defines the basic sizes of the useful data payloads to be encoded in relation with the selected modulation type and encoding rate and concatenation rule.

Table 320—Useful data payload for a FEC block

	QP	SK	16 Ç	QAM		64 QAM	[
Encoding rate	R=1/2	R=3/4	R=1/2	R=3/4	R=1/2	R=2/3	R=3/4
Data payload	6						
(bytes)		9					
	12		12				
	18	18		18	18		
	24		24			24	
		27					27
	30						
	36	36	36	36	36		

• Notice from Table 318:

- O QPSK $\frac{1}{2}$ has a j = 6, that is why there are 6 different payloads for this modulation and rate in Table 320.
- O QPSK $\frac{3}{4}$ has a j = 4, that is why there are 4 different payloads for this modulation and rate in Table 320.

• Notice from Table 320:

- OPSK $\frac{1}{2}$ with 6 bytes data payload will be encoded to 12 bytes and modulated to 12/2 = 6 pairs (1 slot).
- O QPSK $\frac{1}{2}$ with 12 bytes data payload will be encoded to 24 bytes and modulated to 24/2 = 12 pairs (2 slots).
- O QPSK $\frac{1}{2}$ with 36 bytes data payload will be encoded to 72 bytes and modulated to 72/2 = 36 pairs (6 slots).
- OPSK $\frac{3}{4}$ with 9 bytes data payload will be encoded to $9 \times \frac{4}{3} = 12$ bytes and modulated to 6 pairs (1 slot).
- OPSK $\frac{3}{4}$ with 36 bytes data payload will be encoded to $36 \times \frac{4}{3} = 48$ bytes and modulated to 24 pairs (4 slots).

D. RTL Implementation Requirements

Using randomized data from Example 1 with coding rate = ½ and block size of 12 bytes (96 bits):

- For every input bit there is a corresponding 2 output bits X and Y coming out sequentially. For a block of a sequential 96 bits there is a corresponding 192 sequential output bits at double the input data rate.
- This block require two input clocks
 - A 50 MHz clock for the incoming data are serial data at 50 MHz
 - o A 100 MHz output data has double the data rate and they will be at 100 MHz
 - O Use a PLU to generate a 00 MHz clock from a 50 MHz reference clock.
- Use finite state machine to implement the required tail-biting.
- The code should be designed for a multiple input blocks each of 96 bits.

E. Testbench gold data

- Randomized Data (Hex):
 - o 55 8A C4 A5 3A 17 24 E1 63 AC 2B F9
- Convolutional Encoded Data (Hex):
 - O 28 33 E4 8D 39 20 26 D5 B6 DC 5E 4A F4 7A DD 29 49 4B 6C 89 15 13 48 CA

	802.1	16 e														
	Conv	olutiona	al Codin	g, Ta	ail-B	Biting	3									•
	No	Data in	Data	Sh	ift R	egis	ter		_	Ŷ	(X)	Da	ata c	out		HEX
	-110	HEX	in	1	2	3	4	5	6	<u></u>		ىر م		out -		11127
	1			Ι	0	0	1	1	1	0	0/	0	0	1	0	2
	2	5	1_	0	1	0	0	1	1	0	1		•	-	Ŭ	_
	3		0	1_	0	1	0	0	1	0	1	1	0	0	0	8
	4		1	0	1	0	Γ	0	0	0	0					
	5		0	1	0	1	0	1	0	0	0	0	0	1	1	3
	6	5	1	0	1	0	1	0	1	1	1					
	7		0	$\frac{1}{0}$	0	1	0	1	0	0	0	0	0	1	1	3
	8		1	0	1	0	1	0	1	1	1					
	9		1	1	0	1	0	1	0	1	1	1	1	1	0	Е
	10	8	0	1	1	0	1	0	1	0	1					
	11		0	0	1	1	0	1	0	1	0	0	1	0	0	4
-	12		0	0	0	1	1	0	1	0	0					
	13		1	0	0	0	1	1	0	0	1	1	0	0	0	8
	14	A	0	1	0	0	0	1	1	0	0					
	15		1	0	1	0	0	0	1	1	1	1	1	0	1	D
	16		0	1	0	1	<u>0</u> 1	0	0	1	0					
	17		1	0		0		-	0	0	0	0	0	1	1	3
	18 19	C	1 0	1	0 1	1	0	1	0	1	1					
	20		$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	1		1	0	1	0	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	1	0	0	1	9
ł	21		0	0	$\frac{1}{0}$	1	<u>0</u> 1	$\frac{1}{0}$	0	0	0					
	22		1	0	0	0	1	1	0	0	1	0	0	1	0	2
	23	4	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	1	0	0	0	1	1	0	0					
	24		$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	1	0	0	0	1	0	0	0	0	0	0	0
-	25		1	0	0	1	0	0	0	0	0					
	26		0	1	0	0	1	0	0	0	1	0	0	1	0	2
	27	A	1	0	1	0	0	1	0	1	0					
	28		0	1	0	1	0	0	1	0	1	0	1	1	0	6
}	29		0	0	1	0	1	0	0	1	1					_
	30	_		$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0	1	0	1	0	1	0	1	1	0	1	D
	31	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$		1	0	0	1	0	1	1	0		_	6	_	_
	32		1	0	1	0	0	1	0	1	0	0	1	0	1	5

Laa	ı	La	L	0		0	0		L		ı				I
33		0	1	0	1	0	0	1	0	1	1	0	1	1	В
34	3	0	0	1	0	1	0	0	1	1					
35		1	0	0	1	0	1	0	1	0	0	1	1	0	6
36		1	1	0	0	1	0	1	0	1				Ū	Ů
37		1	1	1	0	0	1	0	1	1	1	1	0	1	D
38	A	0	1	1	1	0	0	1	1	0	1	1	U	1	
39	A	1	0	1	1	1	0	0	1	1	1	1	0	0	C
40		0	1	0	1	1	1	0	0	0	1	1	U	U	
41		0	0	1	0	1	1	1	1	0		1	Λ	1	_
42	,	0	0	0	1	0	1	1	1	0	0	1	0	1	5
43	1	0	0	0	0	1	0	1	1	1				0	-
44		1	0	0	0	0	1	0	0	1	1	1	1	0	Е
45		0	1	0	0	0	0	1	1	0					
46		1	0	1	0	0	0	0	0	0	0	1	0	0	4
47	7	1	1	0	1	0	0	0	0	1					
48		1	1	1	0	1	0	0	0	1	1	0	1	0	Α
49	-	0	1	1	1	0	1	0	1	1					
50		$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	1	1	1	0	1	1	1	1	1	1	1	F
51	2	1	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0	1	1	1	0	1	0					
									l		0	1	0	0	4
52		0	1	0	0	1	1	1	0	0					
53		0	0			0		1	1	0	0	1	1	1	7
54	4	1	0	0	1	0	0	1	1	1					
55		0	1	0	0	1	0	0	0	1	1	0	1	0	Α
56		0	0	1	0	0	1	0	0	1					
57		1	0	0	1	0	0	1	1	1	1	1	0	1	D
58	E	1	1	0	0	1	0	0	1	0	1	•	Ů	•	
59		1	1	1	0	0	1	0	1	1	1	1	0	1	D
60		0	1	1	1	0	0	1	1	0	1	1	U	1	D
61		0	0	1	1	1	0	0	0	0	0	0	1	0	2
62	1	0	0	0	1	1	1	0	0	1	0	U	1	U	2
63	1	0	0	0	0	1	1	1	0	1	1	Λ	Λ	1	
64		1	0	0	0	0	1	1	1	0	1	0	0	1	9
65		0	1	0	0	0	0	1	1	0	_	1	0	^	4
66		1	0	1	0	0	0	0	0	0	0	1	0	0	4
67	6	1	1	0	1	0	0	0	0	1		^	^		
68		0	1	1	0	1	0	0	1	0	1	0	0	1	9
69		0	0	1	1	0	1	0	1	0					
70		0	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0	1	1	0	1	0	0	0	1	0	0	4
71	3	1	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0	0	1	1	0	0	1					
72		1	1	0	0	0	1	1	1	1	1	0	1	1	В
73		1	1	1	0	0	0	1	1	0					
			1	1	1						0	1	1	0	6
74	A	0				0	0	0	0	1				•	
75		1	0	1	1	1	0	0	1	1	1	1	0	0	С
76		0	1	0	1	1	1	0	0	0					
77		1	0	1	0	1	1	1	0	1	1	0	0	0	8
78	С	1	1	0	1	0	1	1	0	0		-	-		_
79	-	0	1	1	0	1	0	1	0	1	1	0	0	1	9
80		0	0	1	1	0	1	0	1	0				-	_

_	0	$0 \\ 0$	$0 \\ 0$	1 0	1 1	0 1	1	0	0	0	0	0	1	1
2	1 0	0 1	0	0 0	0 0	1	1	1 1	0	0	1	0	1	5
D	1 0	0	1 0	0	0	0	0	0 1	0	0	0	0	1	1
В	1	0	1 0	0 1	1 0	0 1	0	0	0 1	0	0	1	1	3
Г	1	1 1	1 1	0 1	1 0	0	1 0	1 0	0	0	1	0	0	4
r i	1	1 1	1 1	1 1	1 1	0 1	1	0	1 0	1	0	0	0	8
0	1 0	1 1	1	1	1	1	1	1 0	1 0	1	1	0	0	С
9	0	0	1	1	1	1	1 1 J	0	1	1	0=	1	0	A
	2 B F	2	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	B	B	B	B 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 0	B 1	B 1	2	2	2	2

5. Interleaving⁶

- All encoded data bits shall be interleaved by a block interleaver with a block size corresponding to the number of coded bits per the encoded block size N_{cbps} .
- The interleaver is defined by a two step permutation.
 - o The first ensures that adjacent coded bits are mapped onto nonadjacent subcarriers.
 - The second permutation insures that adjacent coded bits are mapped alternately onto less or more significant bits of the constellation, thus avoiding long runs of lowly reliable bits.
- Let N_{cpc} be the number of coded bits per subcarrier.
 - o That is: 2, 4, or 6 for QPSK, 16-QAM or 64-QAM, respectively.
- Let $s = N_{\rm cpc}/2$.
- Within a block of N_{cbps} bits at transmission, let:
 - \circ k be the index of the coded bit before the first permutation,
 - \circ m_k be the index of that coded bit after the first and before the second permutation,
 - \circ j_k be the index after the second permutation, just prior to modulation mapping, and
 - o d be the modulo used for the permutation.
- The first permutation is defined by Equation (130):

$$m_k = \frac{N_{\text{cbps}}}{d} \cdot k_{\text{mod}(d)} + \left| \frac{k}{d} \right|; \quad k = 0, 1, ..., N_{\text{cbps}} - 1, \quad d = 16$$
 (130)

• The second permutation is defined by Equation (131):

$$j_k = s \cdot \left\lfloor \frac{m_k}{s} \right\rfloor + \left(m_k + N_{\text{cbps}} - \left\lfloor \frac{d \cdot m_k}{N_{\text{cbps}}} \right\rfloor \right)_{\text{mod}(s)}; \quad k = 0, 1, \dots, N_{\text{cbps}} - 1, \quad d = 16$$

(131)

A. RTL Implementation

• Refer to Example 1.

$$N_{\text{cbps}} = 192 \text{ bits, } N_{\text{cpc}} = 2 \text{ bits, } s = 1.$$

$$m_k = \frac{192}{16} \cdot k_{\text{mod}(16)} + \left\lfloor \frac{k}{16} \right\rfloor = 12 \cdot k_{\text{mod}(16)} + \left\lfloor \frac{k}{16} \right\rfloor; \qquad k = 0,1,...,191.$$

$$j_k = \left\lfloor m_k \right\rfloor + \left(m_k + 192 - \left\lfloor \frac{16 \cdot m_k}{192} \right\rfloor \right)_{\text{mod}(1)} = m_k; \qquad k = 0,1,...,191.$$

• Notice that here $j_k = m_k$. That is only one permutation.

⁶ 8.4.9.3

B. Testbench Gold data

802.16 e	
Bit Interleaving, $s = 1$	

Bit I	nterleav	ing, s =	: 1					
No	Data in HEX	Data in DEC	Data in	kmod16	mk	mk	Data out	HEX
0 1 2 3	2	2	0 0 1 0	0 1 2 3	0 12 24 36	0 1 2 3	0 1 0 0	4
4 5 6 7	8	8	1 0 0	4 5 6 7	48 60 72 84	4 5 6 7	1 0 1	В
8 9 10 11	3	3	0 0 1 1	8 9 10 11	96 108 120 132	8 9 10 11	0 0 0	0
12 13 14 15	3	3	0 0 1 1	12 13 14 15	144 156 168 180	12 13 14 15	0 1 0 0	4
16 17 18 19	Е	14	1 1 1 0	0 1 2 3	1 13 25 37	16 17 18 19	0 1 1 1	7
20 21 22 23	4	4	0 1 0 0	4 5 6 7	49 61 73 85	20 21 22 23	1 1 0 1	D
24 25 26 27	8	8	1 0 0 0	8 9 10 11	97 109 121 133	24 25 26 27	1 1 1 1	F
28 29 30 31	D	13	1 1 0 1	12 13 14 15	145 157 169 181	28 29 30 31	1 0 1 0	A
32 33 34 35	3	3	0 0 1 1	0 1 2 3	2 14 26 38	32 33 34 35	0 1 0 0	4
36 37 38 39	9	9	1 0 0 1	4 5 6 7	50 62 74 86	36 37 38 39	0 0 1 0	2
40	2	2	0	8	98	40	1	F

				i				
41			0	9	110	41	1	
42			1	10	122	42	1	
43			0	11	134	43	1	
44			0	12	146	44	0	
45		١,	0	13	158	45	0	١,
46	0	0	0	14	170	46	1	2
47			0	15	182	47	0	
48			0	0	3	48	1	
49	_	_	0	1	15	49	0	
50	2	2	1	2	27	50	1	A
51			0	3	39	51	0	
52			0	4	51	52	0	
53			1	5	63	53	1	
54	6	6	1	6	75	54	0	5
55			$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	7	87	55	1	
56			1	8	99	56	1	
57			1	9	111	57		
	D	13					1	D
58			0 1	10	123	58	0	
59				11	135	59	1	
60			0	12	147	60	0	
61	5	5	1	13	159	61	1	5
62			0	14	171	62	0	
63			1	15	183	63	1	
64			1	0	4	64	1	
65	В	11	0	1	16	65	1	F
66			1	2	28	66	1	
67			1	3	40	67	1	
68			0	4	52	68	0	
69	6	6	1	5	64	69	1	6
70			1	6	76	70	1	"
71			0	7	88	71	0	
72			1	8	100	72	0	
73	D	13	1	9	112	73	0	1
74	1	13	0	10	124	74	0	1
75			1	11	136	75	1	
76			1	12	148	76	1	
77		12	1	13	160	77	1	
78	С	12	0	14	172	78	0	C
79		İ	0	15	184	79	0	İ
80			0	0	5	80	0	
81	ا ۔	_ ا	1	1	17	81	0	
82	5	5	0	2	29	82	0	0
83			1	3	41	83	0	
84			1	4	53	84	0	
85			1	5	65	85	0	
86	Е	14	1	6	77	86	1	2
87			$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	7	89	87	0	
88	4	4	0	8	101	88	0	1
00		<u> </u>	J	1 0	101	1 00	I	_ <u></u>

90 91 0 10 125 90 0 1 137 91 1 1 1 137 91 1 1 1 1 137 91 1 1 1 1 14 173 94 1 A 4 1 1 14 173 94 1 A 4 1 1 14 173 94 1 A 4 4 1	89			1	9	113	89	0	
91									
92 A 10 1 12 149 92 1 A A 10 13 161 93 0 A A 94 1 A 98 0 A 1					1				
93 94 95 A 10 0 13 161 93 0 A 96 97 98 99 F 15 1 0 6 96 0 97 1 5 98 0 99 1 3 42 99 1 <t< td=""><td>92</td><td></td><td></td><td></td><td>12</td><td></td><td>92</td><td></td><td></td></t<>	92				12		92		
94 A 10 1 14 173 94 1 A 96 1 0 15 185 95 0 97 F 15 1 1 18 97 1 5 98 99 1 3 42 99 1 5 100 4 4 54 100 1 1 1 100 1<	93			0	13	161	93	0	
95 0 15 185 95 0 96 1 1 0 6 96 0 97 F 15 1 1 18 97 1 5 99 1 3 42 99 1 5 99 1 100 1 0 4 54 100 1 0 1	1	A	10				l		A
96 1 1 0 6 96 0 0 1 1 18 97 1 5 98 0 1 1 18 97 1 5 98 0 1 3 42 99 1 <td< td=""><td>1</td><td></td><td></td><td></td><td></td><td></td><td>l</td><td></td><td></td></td<>	1						l		
97 98 99 F 15 1 1 18 2 97 30 30 42 1 5 100 101 102 103 0 4 54 6 100 100 1 1 1 5 66 78 102 103 103 104 105 106 1 1 5 66 78 102 103 104 105 106 1 1 1 1 104 105 106 0 7 90 103 106 107 104 107 0 8 102 104 106 107 104 107 0 1 104 106 107 0 1 106 107 106 11 10 126 106 107 106 106 107 1 1 10 10 11 <td>\vdash</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td> </td> <td></td> <td></td>	\vdash						 		
98 F 15 1 2 30 98 0 3 100 101 4 4 54 100 1 100 1 101 4 4 1 5 66 101 0 8 103 0 7 90 103 0 104 0 104 0 104 0 104 0 104 0 103 0 104 104 104 104 104 104 104 104 104 104 <td></td> <td></td> <td></td> <td></td> <td>1</td> <td></td> <td></td> <td></td> <td></td>					1				
99		F	15						5
100									
101									
102				•					
103		4	4	•			i		8
104 105 7 7 1 9 114 105 1 5 106 107 1 1 10 126 106 0 1 106 0 1 106 0 1 106 0 1 106 0 1 100 10 1 10 11 11 10 10 11 11 10 10 11					1				
105 7 7 1 9 114 105 1 5 107 1 1 10 126 106 0 5 108 109 1 12 150 108 0 1 109 10 0 13 162 109 0 1 111 1 14 174 110 0 1 111 1 14 174 110 0 1 112 1 1 1 110 0 1 <td< td=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></td<>									
106		_	_						_
107		1	1		10				5
109 110 A 10 1 14 174 110 0 1 14 174 110 0 1 111 1 111 112 113 1 113 1 114 1 1 10 113 1 1 112 1 113 114 D 13 1 1 1 1 1 1 1 1 10 1 1 1 1 113 1 1 115 1 3 43 115 0 1 4 55 116 1 1 17 0 9 116 117 D 13 1 5 67 117 0 9 118 0 9 118 0 9 119 1 7 91 119 1 1 10 127 122 1 1 14 139 123 0 1 15 124 0 1 12 151 124 0 1 12 151 124 0 1 12 151 124 0 1 12 151 124 0 1 12 151 124 0 1 15 187 127 1 1 15 187				1	11		107	1	
110 A 10 1 14 174 110 0 1 111 111 0 15 186 111 1 1 112 113 1 1 0 7 112 1 1 113 1 1 1 19 113 1 E 115 1 3 43 115 0 1<	108			1	12	150	108	0	
110 1 14 174 110 0 111 0 15 186 111 1 112 1 0 7 112 1 113 1 1 19 113 1 114 1 115 1 3 43 115 0 1 114 1 1<	109		10	0	13	162	109	0	1
112 1 0 7 112 1 1 1 19 113 1 1 19 113 1 113 1 1 113 1 114 1 1 1 114 1 1 1 1 114 1	110	A	10	1	14	174	110	0	1
113 D 13 1 1 19 113 1 E 115 1 3 43 115 0 116 1 4 55 116 1 117 D 13 1 5 67 117 0 118 D 13 0 6 79 118 0 119 1 7 91 119 1 120 0 8 103 120 1 121 2 0 9 115 121 0 122 1 10 127 122 1 A 123 0 11 139 123 0 124 1 12 151 124 0 125 9 0 13 163 125 0 127 1 15 187 127 1 128 0 </td <td>111</td> <td></td> <td></td> <td>0</td> <td>15</td> <td>186</td> <td>111</td> <td>1</td> <td></td>	111			0	15	186	111	1	
114 D 13 0 2 31 114 1 E 116 1 4 55 116 1	112			1	0	7	112	1	
114 1 3 43 114 1 116 1 4 55 116 1 117 118 0 6 79 118 0 119 1 7 91 119 1 120 0 8 103 120 1 121 2 0 9 115 121 0 122 2 1 10 127 122 1 A 123 0 11 139 123 0 124 1 12 151 124 0 125 9 0 13 163 125 0 126 9 0 14 175 126 1 127 1 15 187 127 1 128 0 0 8 128 0 130 4 1 1 20 129 0 131 0 3 44 131 0	113	D	13	•			113		E
116 117 D 13 1 4 55 116 1 9 119 13 1 5 67 117 0 9 119 1 7 91 118 0 9 120 0 8 103 120 1 120 1 121 2 2 0 9 115 121 0 1 122 1 10 127 122 1 A 123 0 11 139 123 0 124 1 12 151 124 0 125 9 9 0 13 163 125 0 126 1 15 187 127 1 1 128 1 1 20 130 0 0 131 0 3 44 131 0 0 132 <td>114</td> <td>שו</td> <td>13</td> <td>0</td> <td>2</td> <td></td> <td>114</td> <td>1</td> <td>L</td>	114	שו	13	0	2		114	1	L
117 118 119 D 13 1 5 67 79 									
118 119 D 13 1 0 1 6 7 91 79 118 119 118 119 0 119 9 119 120 121 122 122 0 2 2 9 2 1 2 1 1 2 1 1 1 2 1 2 1 1 2 1 1 2 1 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 4 									
118 1 7 91 118 0 1 <td></td> <td>D</td> <td>13</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>9</td>		D	13						9
120 0 8 103 120 1 121 0 115 121 0 121 0 1 121 0 A 122 1 A 123 0 1 123 0 1 123 0 1 123 0 1 124 0 1 124 0 1 124 0 1 125 0 1 126 1 3 126 1 1 127 1 1 127 1 1 127 1 1 127 1 1 127 1 1 128 0 129 0 130 0 130 0			13						
121 2 2 0 9 115 121 0 A 123 0 11 139 123 0 A 124 1 12 151 124 0 124 0 125 9 9 0 13 163 125 0 3 126 127 1 15 187 127 1 127 1 128 0 0 8 128 0 0 128 0 0 128 0 0 128 0 0 0 130 0 0 0 130 0									
122 2 1 10 127 122 1 A 123 0 11 139 123 0 1 124 125 1 12 151 124 0 125 9 9 0 13 163 125 0 126 1 15 187 126 1 3 127 1 15 187 127 1 128 0 0 8 128 0 129 1 1 20 129 0 130 0 3 44 131 0 131 0 3 44 131 0 132 1 4 56 132 1 133 9 0 5 68 133 0 134 0 1 7 92 135 1									
122 1 10 127 122 1 123 0 11 139 123 0 124 1 12 151 124 0 125 9 9 0 13 163 125 0 126 1 15 187 126 1 127 1 128 0 0 8 128 0 128 0 128 0 129 0 0 132 130 0 0 132 130 0 0 131 0 0 132 1 0 132 1 132 1 132 1 132 1 133 0 9 134 0 9 134 0 9 134 0 134 0 134 0 134 0 134 0 134 0 135 1 135 1 135 1 135		2	2		1				A
124 1 12 151 124 0 125 0 13 163 125 0 125 0 125 0 125 0 125 0 126 1 125 0 126 1 126 1 126 1 127 1 127 1 127 1 127 1 128 0 128 0 128 0 128 0 129 0 0 130 0 0 130 0 0 130 0 0 130 0 0 130 0 0 0 130 0 0 130 0 0 0 130 0 0 0 130 0 0 130 0 0 130 0 0 0 130 0 0 130 0 0 130 0 0 0 130 0 0 0 0 0 0 <td>1</td> <td></td> <td></td> <td>i</td> <td>i</td> <td>i i</td> <td></td> <td></td> <td></td>	1			i	i	i i			
125 9 9 0 13 163 125 0 3 127 1 15 187 126 1 3 128 0 0 8 127 1 129 130 1 1 20 129 0 131 0 3 44 131 0 132 1 4 56 132 1 133 1 4 56 132 1 133 1 4 56 133 0 9 134 0 6 80 134 0 9 135 1 7 92 135 1									
126 9 9 0 14 175 126 1 3 127 1 15 187 127 1 128 0 0 8 128 0 129 1 1 20 129 0 130 0 2 32 130 0 131 0 3 44 131 0 132 1 4 56 132 1 133 1 4 56 133 0 134 9 0 6 80 134 0 135 1 7 92 135 1	1								
127 1 15 187 127 1 128 0 0 8 128 0 129 1 1 20 129 0 130 0 2 32 130 0 131 0 3 44 131 0 132 1 4 56 132 1 133 9 9 0 5 68 133 0 9 134 9 0 6 80 134 0 9 135 1 7 92 135 1 9	1	9	9	i	i		i		3
128 0 0 8 128 0 0 129 130 1 1 20 129 0 0 131 0 2 32 130 0 0 132 1 4 56 132 1 133 1 4 56 132 1 134 9 0 6 80 134 0 135 1 7 92 135 1									
129 4 4 1 1 20 129 0 0 131 0 3 44 131 0 0 132 1 4 56 132 1 133 9 9 0 5 68 133 0 134 9 0 6 80 134 0 135 1 7 92 135 1									
130 4 4 0 2 32 130 0 0 131 0 3 44 131 0 0 132 133 9 9 0 5 68 132 1 134 9 0 6 80 134 0 9 135 1 7 92 135 1				•	1				
131 0 3 44 131 0 132 1 4 56 132 1 133 9 9 0 5 68 133 0 134 9 0 6 80 134 0 135 1 7 92 135 1		4	4						0
132 133 9 9 1 4 56 132 1 133 134 134 0 134 0 134 0 135 1 7 92 135 1 9				•	3				
133 9 9 0 5 68 133 0 9 135 1 7 92 135 1 9									
134 9 9 0 6 80 134 0 9 135 1 7 92 135 1									
135 1 7 92 135 1		9	9	•					9
				i	1				
	136	4	4	0	8	104	136	1	A

138	137	I	İ	l 1	ا م	114	ı	127	۱۸	I
139										
140										
141 142 142 143 B 11 0 13 13 164 176 142 1 142 1 143 0 141 0 15 188 143 0 142 1 1 145 1 145 1 144 0 144 0 144 0 145 1 145 1 145 1 145 1 145 1 145 1 146 0 144 0 145 1							ŀ			
142 B 11 1 14 176 142 1 2 143 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 144 0 144 144 0 144 147 0 144 147 0 144 147 0 144 147 0 144 147 0 144 147 0 144 <td></td> <td></td> <td></td> <td></td> <td>1</td> <td></td> <td></td> <td></td> <td></td> <td></td>					1					
142 1 14 1/6 142 1 1 15 188 143 0 144 0 0 0 9 144 0 144 0 0 0 144 0 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 0 144 144 0 144 0 144 147 0 144 147 0 144 147 0 144 147 0 144 147 0 148 1 144 147 0 144 147 0 1 144 147 0 1 144 147 0 1 144 147 1 1 144 147 1 1 144 144 144 144 144 144 144 144 144		В	11						•	2
144 145 6 6 1 1 1 21 144 0 4 147 0 3 45 146 0 4 148 1 4 57 148 1 4 147 0 148 1 4 57 148 1 7 93 151 1 F 150 12 1 5 69 149 1 F 155 148 1									•	_
145 6 6 1 1 2 33 146 0 4 147 0 3 45 147 0 4 148 1 4 57 148 1						_	ļ			
146 6 1 2 33 146 0 4 147 148 1 4 57 148 1 7 148 1 1 14 57 148 1 1 14 149 1 1 148 1 1 14 149 1 1 148 1 1 148 1 1 148 1 1 148 1 1 1 148 1										
146 1 2 33 146 0 147 0 148 1 4 57 148 1 149 1 149 1 148 1 148 1 148 1 148 1 148 1 148 1 148 1 148 1 148 1 148 1 148 1 148 1 148 1 149 1 1 149 1 1 149 1 1 1 149 1 1 1 149 1		6	6	•	1				1	14
148 149 C 12 1 4 57 148 1 149 1 1 149 1 149 1 149 1 1 149 1 1 149 1 1 149 1 1 149 1 1 1 149 1 1 1 149 1									0	'
149 150 150 150 150 150 150 151 150 151 150 151 150 151 150 151 150 151 150 151 150 151 150 151 150 151 150 151 151 150 151 150 151 150 151 150 151 150 150 151 150 151 150 151 150 151 150 151 150 151 150 1						_	ļ			
150	148			1	!	57		148	1	
150	149		12	1	5	69		149	1	 E
152 1 8 105 152 1 153 1 153 1 1 153 1 1 1 153 1 1 153 1 1 153 1 154 0 1 155 1 1 155 1 1 155 1 1 155 1 1 155 1 1 155 1 1 155 1 1 155 1 1 155 1 1 155 1 1 155 1 1 155 1 1 156 0 1 157 1 5 1 157 1 158 0 1 157 1 158 0 1 1 158 0 1 159 1 1 158 0 1 1 1 158 1 159 1 1 1 1 158 1 1 1 1 <t< td=""><td>150</td><td></td><td>12</td><td>0</td><td>6</td><td>81</td><td></td><td>150</td><td>1</td><td>F</td></t<>	150		12	0	6	81		150	1	F
153 8 8 0 9 117 153 1 D 155 0 11 141 155 1 156 1 12 153 156 0 157 158 9 9 0 13 165 157 1 5 159 1 15 189 159 1 158 0 5 160 1 15 189 159 1 5 1 156 0 1 160 1 160 1	151			0	7	93		151	1	
154 8 0 10 129 154 0 D 155 0 11 141 155 1 156 1 12 153 156 0 157 158 0 13 165 157 1 5 159 1 15 189 159 1 15 158 0 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 1 160 1 1 160 1 1 160 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	152			1	8	105	Ī	152	1	
154 8 0 10 129 154 0 D 155 0 11 141 155 1 156 1 12 153 156 0 157 158 0 13 165 157 1 5 159 1 15 189 159 1 158 0 5 160 1 15 189 159 1 5 1 15 189 159 1 1 160 1 160 1 160 1 160 1 160 1 1 160 1 160 1 1 160 1 1 160 1 1 160 1 1 1 160 1 1 1 1 160 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	153			0	9	117	İ	153	1	
155 0 11 141 155 1 156 1 12 153 156 0 157 9 9 0 13 165 157 1 5 158 159 1 155 189 159 1 158 0 1 155 1 158 0 1 159 1 159 1 159 1 159 1 159 1 159 1 159 1 159 1 159 1 159 1 159 1 159 1 159 1 159 1 159 1 150 1 159 1 150 1 150 1 150 1 150 1 150 1 150 1 150 1 150 1 1 150 1 1 150 1 1 1 150 1 1 1 1	154	8	8	0	10		ı	154	0	שן
156 1 12 153 156 0 157 1 5 157 1 5 157 1 5 157 1 5 157 1 5 157 1 5 158 0 1 158 0 1 158 0 1 158 0 1 158 0 1 158 0 1 158 0 1 158 0 1 1 158 0 1 1 158 0 1 159 1 1 1 158 0 1 1 159 1<				0	1		İ	155	1	
157 9 9 0 13 165 157 1 5 159 1 15 189 159 1 160 1 15 189 159 1 160 1 15 189 159 1 160 1 15 189 159 1 160 1 15 189 159 1 161 1 0 1 160 1 162 1 1 0 1 160 1 163 1 3 46 163 0 1 164 1 3 46 163 0 0 165 5 1 5 70 165 0 0 166 1 7 94 167 0 0 1 170 1 1 1 1 1 1 1 1						_	İ			
158 9 9 0 14 177 158 0 5 159 1 15 189 159 1 1 159 1 1 159 1 1 159 1 1 159 1 1 159 1 1 1 160 1 1 1 160 1 1 160 1 <t< td=""><td></td><td></td><td></td><td></td><td></td><td></td><td>ı</td><td></td><td></td><td>İ _</td></t<>							ı			İ _
159 1 15 189 159 1 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 161 0 8 162 0 8 162 0 1 162 0 8 164 0 1 165 0 165 0 0 1 165 0 0 1 165 0 0 1 165 0 0 0 1 165 0 0 0 1 165 0 0 1 166 0 0 1 167 0 0 1 167 0 1 167 0 1 167 0 1		9	9	1	1					5
160 1 0 0 10 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 160 1 161 0 8 161 0 161 0 162 0 162 0 162 0 163 0 163 0 164 0 165 0 165 0 165 0 165 0 165 0 165 0 165 0 165 0 165 0 0 165 0 0 165 0 0 165 0 0 165 0 0 165 0 0 165 0 0 166 0 165 0 0 165 0 0 165 0 0 168 1 167 0 0 170 170 170 170 170 170 170										
161 162 1 1 0 1 22 34 162 0 8 163 1 1 3 46 162 0 0 8 164 165 165 166 166 167 1 0 4 58 164 0 0 0 0 165 0 0 0 0 165 0 0 0 0 165 0 0 0 0 165 0 0 0 0 165 0 0 0 165 0 0 0 165 0 0 0 165 0 0 0 165 0 0 0 166 0 0 0 166 0 0 0 166 0 0 0 167 0 0 0 168 1 169 0 0 0 170 0 170 0 170 0 170 0 170 0 170 0 170 0 171 0 171 0 171 0 172 0 0 172 173 1 0 173 1 174 1 174 1 174 1 174 1 174 1 174 1 174 1 174 1 175 0 175 0 175 0 175 0 175 0 175 0 175 0 175 0 175 0 177 0 177 0 178 1 177						_	t			
162 1 1 0 2 34 162 0 8 163 0 4 58 164 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	ı				i				1	
163 1 3 46 163 0 164 0 4 58 164 0 165 5 1 5 70 165 0 166 1 7 94 167 0 168 0 8 106 168 1 169 1 1 0 9 118 169 0 170 1 1 1 11 142 171 0 171 1 1 11 142 171 0 172 0 12 154 172 0 173 3 3 13 166 173 1 6 173 1 14 178 174 1 1 1 1 1 1 6 1 1 1 6 1 1 1 1 1 1 1 1 1 <td< td=""><td></td><td>1</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td>8</td></td<>		1	1							8
164 0 4 58 164 0 0 0 0 0 165 0 0 0 0 165 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0										
165 5 5 1 5 70 165 0 0 167 1 7 94 167 0 168 0 8 106 168 1 169 1 1 0 9 118 169 0 170 1 1 10 130 170 0 8 171 1 11 142 171 0 170 0 172 0 12 154 172 0 0 173 3 3 13 166 173 1 6 173 1 14 178 174 1 6 175 1 15 190 175 0 176 0 0 11 176 1 177 4 4 1 1 23 177 0 180 0 3 47 179 1 181 1 4 59 180 1						_	ł			
166 3 3 0 6 82 166 0 0 168 169 1 1 0 8 106 168 1 168 1 169 0 168 1 169 0 168 1 169 0 8 170 0 170 170 170 0 170 170 170 170 170 170 170 170 170 170 170 170 170 170 170 170 </td <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>										
167 1 7 94 167 0 168 0 8 106 168 1 169 1 1 0 9 118 169 0 170 1 1 0 10 130 170 0 171 1 1 11 142 171 0 172 0 12 154 172 0 173 3 0 13 166 173 1 6 174 1 15 190 175 0 175 0 176 1 15 190 175 0 1 176 1 176 1 176 1 177 0 1 177 0 1 178 1 1 177 0 1 1 176 1 1 1 1 1 1 1 1 1 1 1		5	5		1					0
168 169 1 1 0 8 106 168 1 169 0 8 170 1 1 0 9 118 169 0 8 171 1 1 11 142 170 0 170 0 172 0 12 154 172 0 172 0 173 1 6 173 1 1 14 178 174 1 6 175 1 15 190 175 0 176 0 0 11 176 1 178 4 4 1 1 23 177 0 B 179 0 3 47 179 1 B 180 1 4 59 180 1 181 8 0 5 71 181 1 182 0 7 95 183 1					1					
169 170 1 1 0 9 118 130 170 0 8 171 1 1 11 11 142 171 0 172 0 172 173 3 3 0 13 166 173 1 1 174 174 1 1 14 178 174 1 1 175 1 1 15 190 175 0 1 176 177 178 1 4 4 1 1 1 23 177 0 1 178 179 1 0 3 47 179 1 1 180 181 8 8 8 0 0 5 71 181 1 1 181 82 0 0 7 95 183 1 0						_	ł			
170 1 1 0 10 130 170 0 8 171 1 1 11 142 171 0 8 172 0 12 154 172 0 172 0 173 1 172 0 0 173 1 6 173 1 6 174 1 1 174 1 1 174 1 1 174 1 1 174 1 1 174 1 1 174 1 1 174 1 1 175 0 1 174 1 1 175 0 1 174 1 1 176 1 1 176 1 1 176 1 1 177 0 1 1 177 0 1 1 177 0 1 1 177 0 1 1 177 0 1 1 178 1 1 1 179 1 1 1 1 1 1 <t< td=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></t<>										
171 1 11 142 171 0 172 0 12 154 172 0 173 3 3 0 13 166 173 1 6 174 1 14 178 174 1 6 175 1 15 190 175 0 176 1 15 190 175 0 177 4 4 1 1 23 177 0 B 178 1 0 2 35 178 1 B 179 0 3 47 179 1 B 180 1 4 59 180 1 B 181 8 8 0 6 83 182 0 D 183 0 7 95 183 1 D		1	1	i	1					8
172 0 12 154 172 0 173 3 0 13 166 173 1 6 174 1 14 178 174 1 6 175 1 15 190 175 0 176 0 0 11 176 1 177 4 1 1 23 177 0 B 179 0 3 47 179 1 B 180 1 4 59 180 1 181 8 0 5 71 181 1 182 0 6 83 182 0 183 0 7 95 183 1	•				1				•	
173 174 3 3 0 13 166 178 174 178 174 						_	ł			
174 3 3 1 14 178 174 1 6 175 1 15 190 175 0 176 0 0 11 176 1 177 4 1 1 23 177 0 178 1 0 2 35 178 1 179 0 3 47 179 1 180 1 4 59 180 1 181 8 0 5 71 181 1 182 8 0 6 83 182 0 183 0 7 95 183 1	ı				i	i i			i	
174 1 14 178 174 1 175 1 15 190 175 0 176 0 0 11 176 1 177 4 1 1 23 177 0 B 178 1 0 2 35 178 1 B 179 0 3 47 179 1 180 1 4 59 180 1 181 8 0 5 71 181 1 182 8 0 6 83 182 0 183 0 7 95 183 1	i .	3	3	1	l					6
176 177 4 4 1 1 23 177 0 1 178 1 1 1 23 177 0 1 179 0 2 35 178 1 179 1 180 1 4 59 180 1 181 8 0 5 71 181 1 182 8 0 6 83 182 0 183 0 7 95 183 1										
177 4 4 1 1 23 177 0 B 178 1 0 2 35 178 1 B 179 0 3 47 179 1 180 1 4 59 180 1 181 8 0 5 71 181 1 182 0 6 83 182 0 183 0 7 95 183 1							ļ			
178 4 4 0 2 35 178 1 B 179 0 3 47 179 1 180 1 4 59 180 1 181 8 0 5 71 181 1 182 8 0 6 83 182 0 183 0 7 95 183 1				l	1					
178 0 2 35 178 1 179 0 3 47 179 1 180 1 4 59 180 1 181 8 0 5 71 181 1 182 8 0 6 83 182 0 183 0 7 95 183 1		4	4							В
180 1 4 59 180 1 181 8 8 0 5 71 181 1 182 8 0 6 83 182 0 183 0 7 95 183 1		'	•							~
181 8 8 0 5 71 181 1 182 0 183 0 7 95 183 1 D						_				
182 8 0 6 83 182 0 D 183 0 7 95 183 1				1					1	
182 0 6 83 182 0 183 0 7 95 183 1		8	8	0				181	1	D
	182	0	0	0	6	83		182	0	ر ا
184 C 12 1 8 107 184 0 1	183	<u> </u>		0		95		<u>18</u> 3	1	
	184	C	12	1	8	107	Ī	184	0	1

185			1	9	119	185	0	
186			0	10	131	186	0	
187			0	11	143	187	1	
188			1	12	155	188	1	
189	.	10	0	13	167	189	1	г
190	A	10	1	14	179	190	1	Е
191			0	15	191	191	0	

6. Modulation⁷

A. Data modulation⁸

- After the repetition block, the data bits are entered serially to the constellation mapper.
- Gray-mapped QPSK and 16-QAM (as shown in Figure 263) shall be supported.
- The constellations (as shown in Figure 263) shall be normalized by multiplying the constellation point with the indicated factor c to achieve equal average power.

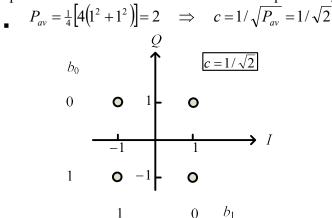


Figure 263a—QPSK constellations

B. RTL Implementation of QPSK

The I and Q points of the QPSK constellation is in Q15 format (16-bit fixed point, MSB as sign bit, 15 fractional bits, and two's complement.)

C. Tesbench Gold Data

• Refer to Example 1 and 6.

802.16 e										
Cons	Constellation Mapping									
No	Int. Data in	Consta Mappi	allation ng	Normaliz Constella Mapping	•					
0	0	山	-1	0.707	-0.707					
$\begin{vmatrix} 2 \\ 3 \end{vmatrix}$	0	1	1	0.707	0.707					
4 5	1 0	-1	1	-0.707	0.707	•				
6 7	1	-1	-1	-0.707	-0.707					
8 9	0	1	1	0.707	0.707					
10	$\begin{vmatrix} \ddot{0} \end{vmatrix}$	1	1	0.707	0.707					

⁷ 8.4.9.4

8 8.4.9.4.2

11	0				
12	0	1	-1	0.707	-0.707
13 14	1 0	-	-	01,01	01,0,
15	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	1	1	0.707	0.707
16 17	0	1	-1	0.707	-0.707
18	1	-1	-1	-0.707	-0.707
19	1				
21	1	-1	-1	-0.707	-0.707
22 23	0	1	-1	0.707	-0.707
24 25	1	-1	-1	-0.707	-0.707
26 27	1	-1	-1	-0.707	-0.707
28 29	1 0	-1	1	-0.707	0.707
30	1	1	1	0.707	0.707
31	0	-1	1	-0.707	0.707
32 33	0	1	-1	0.707	-0.707
34	0	1	1	0.707	0.707
35	0	1	1	0.707	0.707
36	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	1	1	0.707	0.707
38 39	1 0	-1	1	-0.707	0.707
40 41	1 1	-1	-1	-0.707	-0.707
42 43	1 1	-1	-1	-0.707	-0.707
44 45	0 0	1	1	0.707	0.707
46 47	1 0	-1	1	-0.707	0.707
48 49	1 0	-1	1	-0.707	0.707
50	1	-1	1	-0.707	0.707
51 52	0				0.=
53	1	1	-1	0.707	-0.707
54 55	0	1	-1	0.707	-0.707
56	1	1	1	0.707	0.707
57	1	-1	-1	-0.707	-0.707
58 59	0	1	-1	0.707	-0.707
1 27	L				

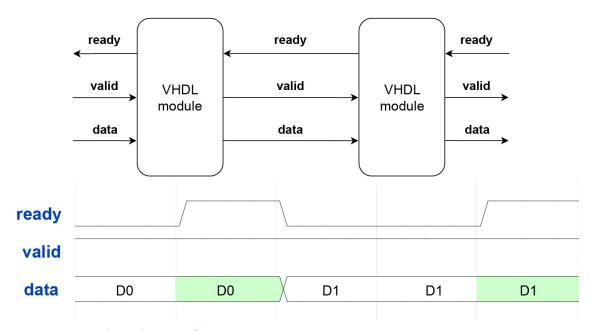
	-				
60 61	0	1	-1	0.707	-0.707
62	0	1	-1	0.707	-0.707
63	1		-	0.707	0.707
64 65	1	-1	-1	-0.707	-0.707
66	1	-1	-1	-0.707	-0.707
67	1				
68 69	0	1	-1	0.707	-0.707
70 71	1 0	-1	1	-0.707	0.707
72	0				
73	0	1	1	0.707	0.707
74 75	0	1	-1	0.707	-0.707
76 77	1	-1	-1	-0.707	-0.707
78	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$				
79	0	1	1	0.707	0.707
80	0	1	1	0.707	0.707
81	0	1	1	0.707	0.707
82	0	1	1	0.707	0.707
83	0	1	1	0.707	0.707
84	0	1	1	0.707	0.707
85	0	1	1	0.707	0.707
86	1	-1	1	-0.707	0.707
87	0				
88 89	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	1	1	0.707	0.707
90	0	1	-1	0.707	-0.707
91	1	_	1	0.707	0.707
92 93	1 0	-1	1	-0.707	0.707
94	1				
95	0	-1	1	-0.707	0.707
96	0	1	-1	0.707	-0.707
97	1	1	-1	0.707	-0.707
98	0	1	-1	0.707	-0.707
99	1	1	-1	0.707	-0./0/
100	1	-1	1	-0.707	0.707
101	0	-1	1	-0.707	0.707
102	0	1	1	0.707	0.707
103	0		•	0.707	0.707
104	0	1	-1	0.707	-0.707
105	1	1	1	0.707	0.707
106	0	1	-1	0.707	-0.707
107	1				
108	0	1	1	0.707	0.707

109	0				
110	0	1	-1	0.707	-0.707
111	1	1	-1	0.707	-0.707
112	1	-1	-1	-0.707	-0.707
113	1				
114	1 0	-1	1	-0.707	0.707
116	1				
117	0	-1	1	-0.707	0.707
118	0	1	-1	0.707	-0.707
119	1	1	-1	0.707	-0.707
120	1	-1	1	-0.707	0.707
121 122	0				
123	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	-1	1	-0.707	0.707
124	0				
125	0	1	1	0.707	0.707
126	1	-1	-1	-0.707	-0.707
127	1	-1	-1	-0.707	-0.707
128	0	1	1	0.707	0.707
129	0				
131	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	1	1	0.707	0.707
132	1	1	1	0.707	0.707
133	0	-1	1	-0.707	0.707
134	0	1	-1	0.707	-0.707
135	1	1	1	0.707	0.707
136 137	1 0	-1	1	-0.707	0.707
137	1				
139	0	-1	1	-0.707	0.707
140	0	1	1	0.707	0.707
141	0	1	1	0.707	0.707
142	1	-1	1	-0.707	0.707
143	0				
144 145	0	1	-1	0.707	-0.707
146	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$			0.505	0.505
147	0	1	1	0.707	0.707
148	1	-1	-1	-0.707	-0.707
149	1	-1	-1	-0.707	-0.707
150	1	-1	-1	-0.707	-0.707
151 152	1				
153	1	-1	-1	-0.707	-0.707
154	0	1	1	0.707	0.707
155	1	1	-1	0.707	-0.707
156	0	1	-1	0.707	-0.707
157	1	1	1	3.707	0.707

158	0	1	-1	0.707	-0.707	
159	1	1	-1	0.707	-0.707	
160	1	-1	1	-0.707	0.707	
161	0	1	1	0.707	0.707	
162	0	1	1	0.707	0.707	
163	0	1	1	0.707	0.707	
164	0	1	1	0.707	0.707	
165	0	1	1	0.707	0.707	
166	0	1	1	0.707	0.707	
167	0	1	1	0.707	0.707	
168	1	-1	1	-0.707	0.707	
169	0	1	1	0.707	0.707	
170	0	1	1	0.707	0.707	
171	0	1	1	0.707	0.707	
172	0	1	-1	0.707	-0.707	
173	1	1	-	01,0,	01,0,	
174	1	-1	1	-0.707	0.707	
175	0			01,0,		
176	1	-1	1	-0.707	0.707	
177	0	_	-	01,0,	01,0,	
178	1	-1	-1	-0.707	-0.707	
179	1					
180	1	-1	-1	-0.707	-0.707	
181	1					
182	0	1	-1	0.707	-0.707	
183	1					
184	0	1	1	0.707	0.707	
185	0					
186	0	1	-1	0.707	-0.707	
187	1					
188	1	-1	-1	-0.707	-0.707	
189	1					
190	1	-1	1	-0.707	0.707	
191	0	_	-			

7. Using the ready/valid handshake9

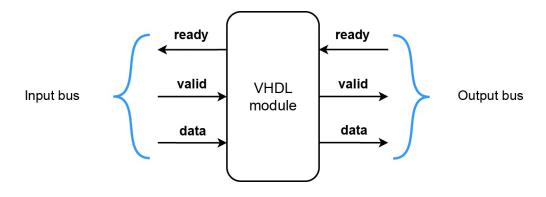
The ready/valid hardware data transfer protocol is simple and ingenious, providing flow control with only two control signals. The rules are straightforward: data transfer only happens when both ready and valid are '1' during the same clock cycle.



A. Receiver and sender interface

Stream processing VHDL modules typically have input and output interfaces. That's because they sit on the data path and do various transformations on the data before passing it on to downstream modules.

The diagram below shows the outline of such a module. Notice that the input and output buses have identical names but with the data directions swapped.



⁹ https://vhdlwhiz.com/how-the-axi-style-ready-valid-handshake-works/

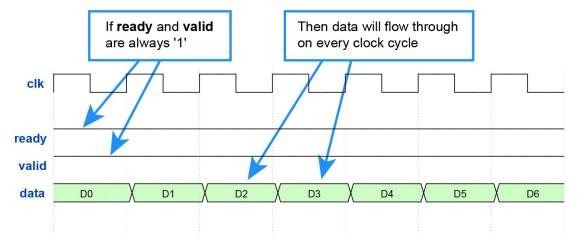
B. Streaming pipeline code example

It shows the entity of a VHDL module corresponding to the diagram above. The input signals are prefixed with in_ and the outputs with out_. Every signal's in/out modes are opposite on the input and output buses, as in the diagram.

```
1
     entity pipeline is
2
       port (
3
         clk : in std_logic;
4
         rst : in std_logic;
5
         -- Input bus
6
7
         in_ready : out std_logic;
8
         in_valid : in std_logic;
9
         in_data : in std_logic_vector(23 downto 0);
10
11
         -- Output bus
12
         out_ready : in std_logic;
13
         out_valid : out std_logic;
         out data : out std logic vector(23 downto 0)
14
15
16
     end pipeline;
```

C. Sending and receiving at full speed

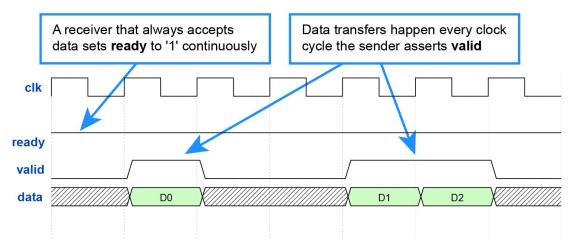
The simplest example I can think of is when ready and valid are '1' continuously. Then, data flows through the interface unhindered, with one transaction on every clock cycle. It's as if there was no flow control.



D. Slow writer and fast reader

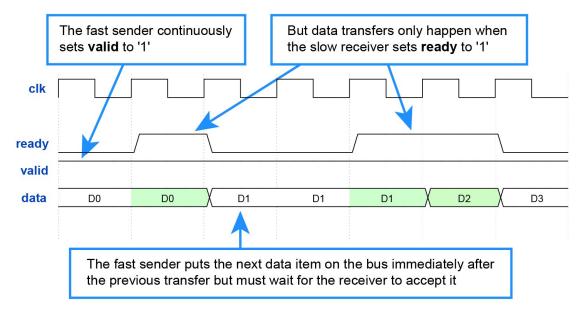
In this example, the ready signal is always '1' while the sender asserts valid occasionally. When implementing a reader module that is guaranteed to accept a data item on every clock cycle, you can simply hardwire ready to always accept data:

```
ready <= '1';
```



E. Fast writer and slow reader

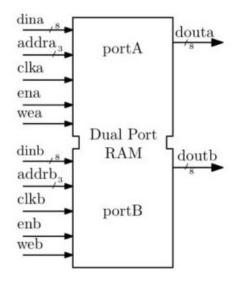
This waveform shows a situation where the reader module is throttling the data rate. We say that the downstream module exerts backpressure when it pauses the data stream like that.



8. Streaming using Pipelining

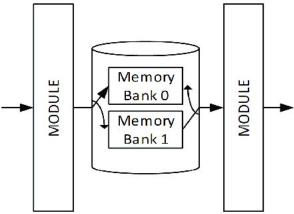
A. Dual-Port Memory

Dual Port Memory (DPR) uses the same memory array. DPR uses two separate ports to access the array. The two ports can be on separate clock domains.



B. Ping-Pong Buffer

Data packets are continuously written to alternating banks. Data packets are continuously read form an opposite banks



Packet #	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	
write to	Bank 0	Bank 1	Bank 0	Bank 1	Bank 0	Bank 1	Bank 0	Bank 1	Bank 0	Bank 1	
Packet #		N1	N2	N3	N4	N5	N6	N7	N8	N9	N10
Read from		Bank 0	Bank 1	Bank 0	Bank 1	Bank 0	Bank 1	Bank 0	Bank 1	Bank 0	Bank 1

9. Project Requirements

- 1. The "Randomizer" block has already been implemented in Lab 2. Each student within a group will be responsible for the implementation of all blocks, the top level that integrates all four including the randomizer, end-to-end testbench verification, and hardware validation.
- 2. Before RTL coding, design each block and provide a design document that include the following:
 - a. A table describing the pin list of every block
 - b. State diagrams for the cases that requires finite state machines
 - c. Timing diagram to explain the interrelationships between signals

- 3. Implement each block separately and write the corresponding testbench that verifies the functionality of each block. You may want to check the output of every block against the tables provided in the description of each block.
- 4. When all blocks are designed and verified, instantiate all blocks in a top level RTL and end-to-end testbench.
- 5. The whole system has to be implemented and validated on the DE0-CV Board.

10.Evaluation Plan

- 1. Each student will be evaluated individually for the progress of the project. The following deliverables will be evaluated:
 - a. Phase I: Block Design and RTL Implementation
 - i. Design document
 - ii. RTL code of the individual blocks
 - iii. Testbench of the individual blocks
 - iv. Simulation results
 - v. Demo and first evaluation
 - b. Phase II: Integration of all blocks and Testbench Verification
 - i. End-to-end testbench of the whole system
 - ii. Top level RTL code that encompasses all blocks
 - iii. End-to-end Simulation results
 - iv. Demo and second evaluation
 - c. Phase II: Hardware Verification
 - i. Successful Synthesis
 - ii. Successful place and route
 - iii. End-to-end hardware validation using DE0-CV Board.
 - iv. Demo and third evaluation