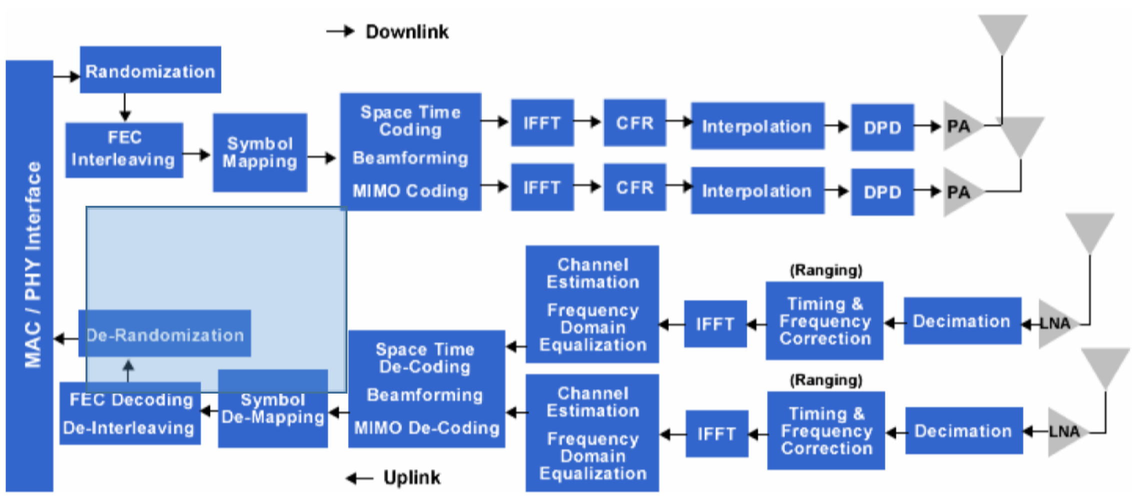
Project Specifications

WiMax PHY—Channel Coding

1. Introduction

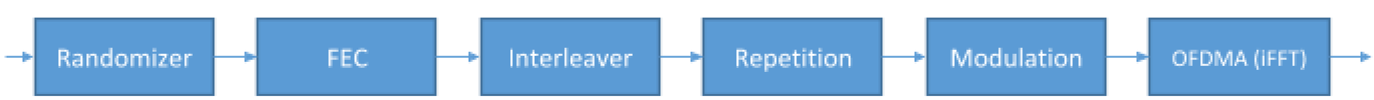
In this project, you are required to implement part of the PHY layer of a WiMax system; namely "Channel Coding" (QPSK only) as described in section 8.4.9 in WiMax Standard (IEEE Std 802.16-2007). This part of the standard is explained in this document with some numeric examples. WiMax PHY including "Channel 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 Coding[1]

- 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)



3.

[1] See section (8.4.9)

Randomizer[2]

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.

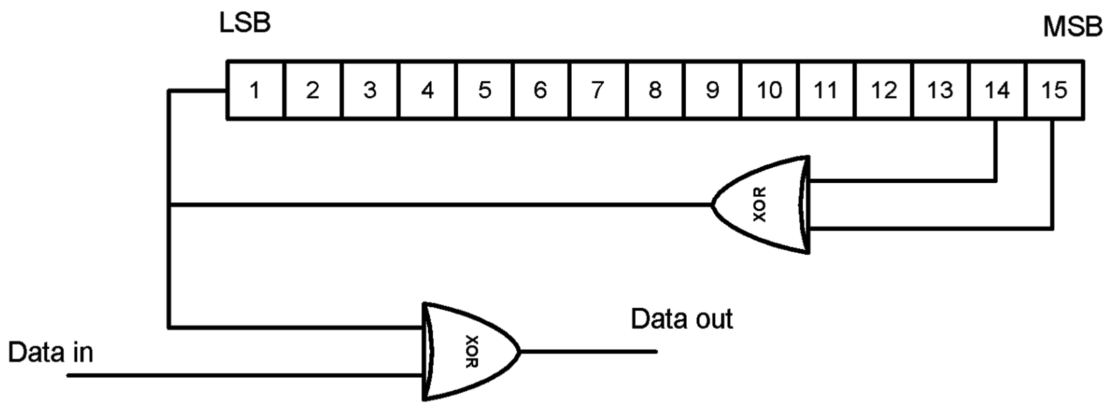
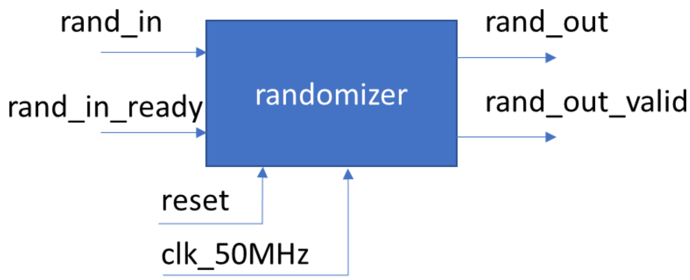


Figure 253—PRBS generator for data randomization

- The randomizer is initialized with the vector:
    - [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

[2] Section 8.4.9.1 in the standard



C. Testbench Gold Data:

- Number of symbols Ns = 2 symbols
- Symbol size = 48 bits
- Block size = Ns * 48 = 96 bits
- Input Data (Hex):
  - AC BC D2 11 4D AE 15 77 C6 DB F4 C9
- Randomized Data (Hex):
  - 55 8A C4 A5 3A 17 24 E1 63 AC 2B F9

802.16 e

Ns = 2 slots 96 bits Initialized vector 011 0111 0001 0101

No	Shift Register															XOR #1	Data in		Data out	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		0x	0b	0b	0x
1	0	1	1	0	1	1	1	0	0	0	1	0	1	0	1	1	A	1	0	5
2	1	0	1	1	0	1	1	1	0	0	0	1	0	1	0	1		0	1	
3	1	1	0	1	1	0	1	1	1	0	0	0	1	0	1	1		1	0	
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	C	1	0	5
6	1	1	1	1	1	0	1	1	0	1	1	1	0	0	0	0		1	1	
7	0	1	1	1	1	1	0	1	1	0	1	1	1	0	0	0		0	0	
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	B	1	1	8
10	0	1	0	0	1	1	1	1	1	0	1	1	0	1	1	0		0	0	
11	0	0	1	0	0	1	1	1	1	1	0	1	1	0	1	1		1	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	C	1	1	A
14	0	1	1	0	0	1	0	0	1	1	1	1	1	0	1	1		1	0	
15	1	0	1	1	0	0	1	0	0	1	1	1	1	1	0	1		0	1	
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	D	1	1	C
18	0	0	1	1	0	1	1	0	0	1	0	0	1	1	1	0		1	1	
19	0	0	0	1	1	0	1	1	0	0	1	0	0	1	1	0		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	2	0	0	4

22	0	1	0	0	0	0	1	1	0	1	1	0	0	1	0	1		0	1	
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	1	0	1	A
26	1	0	1	1	0	1	0	0	0	0	1	1	0	1	1	0		0	0	
27	0	1	0	1	1	0	1	0	0	0	0	1	1	0	1	1		0	1	
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	1	0	0	5
30	0	1	1	0	1	0	1	1	0	1	0	0	0	0	1	1		0	1	
31	1	0	1	1	0	1	0	1	1	0	1	0	0	0	0	0		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	4	0	0	3
34	0	0	0	1	0	1	1	0	1	0	1	1	0	1	0	1		1	0	
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	A
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		1	0	
41	1	1	1	0	1	1	1	0	0	0	1	0	1	1	0	1	A	1	0	1
42	1	1	1	1	0	1	1	1	0	0	0	1	0	1	1	0		0	0	
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	E	1	0	7
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		1	1	
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	1	0	0	2
50	0	1	0	0	1	1	1	0	1	1	1	1	0	1	1	0		0	0	
51	0	0	1	0	0	1	1	1	0	1	1	1	1	0	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	5	0	0	4
54	0	1	1	0	0	1	0	0	1	1	1	0	1	1	1	0		1	1	
55	0	0	1	1	0	0	1	0	0	1	1	1	0	1	1	0		0	0	
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	7	0	1	E
58	1	1	0	0	0	1	1	0	0	1	0	0	1	1	1	0		1	1	
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	0	
61	1	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0	7	0	0	1
62	0	1	0	0	1	1	0	0	0	1	1	0	0	1	0	1		1	0	
63	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	C	1	0	6
66	1	0	1	1	0	1	0	0	1	1	0	0	0	1	1	0		1	1	
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		0	0	

69	0	1	0	1	0	1	1	0	1	0	0	1	1	0	0	0	6	0	0	3
70	0	0	1	0	1	0	1	1	0	1	0	0	1	1	0	1		1	0	
71	1	0	0	1	0	1	0	1	1	0	1	0	0	1	1	0		1	1	
72	0	1	0	0	1	0	1	0	1	1	0	1	0	0	1	1		0	1	
73	1	0	1	0	0	1	0	1	0	1	1	0	1	0	0	0	D	1	1	A
74	0	1	0	1	0	0	1	0	1	0	1	1	0	1	0	1		1	0	
75	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1	1		0	1	
76	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1		1	0	
77	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	B	1	1	C
78	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1		0	1	
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	0	
81	1	1	1	0	1	1	1	0	1	0	1	0	0	1	0	1	F	1	0	2
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		1	1	
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	B
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	C	1	1	F
90	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	0		1	1	
91	0	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1		0	1	
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	9	1	1	9
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		0	0	
96	0	0	0	1	1	0	0	1	1	1	1	1	0	1	1	0		1	1	

4.

FEC Encoder[3]

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:
    - j: parameter dependent on the modulation and FEC rate
    - n: floor(number of allocated slots * STC rate/(repetition factor * number of STC layers))
    - $k : \left\lfloor \frac{n}{j} \right\rfloor$
    - 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 n slots

$n > j$	If (n mod j = 0)
	k blocks of j slots
	else
	(k–1) blocks of j slots
	1 block of $\left\lceil \frac{m+j}{2} \right\rceil$ slots
	1 block of $\left\lfloor \frac{m+j}{2} \right\rfloor$ slots

[3] 8.4.9.2

Table 318—Encoding slot concatenation for different allocations and 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.
- $j = 6, \quad n = \left\lfloor \frac{N_s}{R} \right\rfloor = \left\lfloor \frac{2}{1} \right\rfloor = 2 \quad \Rightarrow \quad n \le j$.
  - 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.
- $j = 6, n = \left\lfloor \frac{N_s}{R} \right\rfloor = \left\lfloor \frac{12}{1} \right\rfloor = 12 \quad \Rightarrow \quad n > j, k = \left\lfloor \frac{n}{j} \right\rfloor = \left\lfloor \frac{12}{6} \right\rfloor = 2, m = 0$.
  - 2 block of 6 slots:
    - 2 blocks of $48 \times 6 = 288$ bits (36 bytes).

C.

Convolutional Coding[4]

- 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:

$$G_1 = 171_{OCT}; \quad \text{For } X$$

$$G_2 = 133_{OCT}; \quad \text{For } Y \qquad (125)$$

- The generator is depicted in Figure 255.

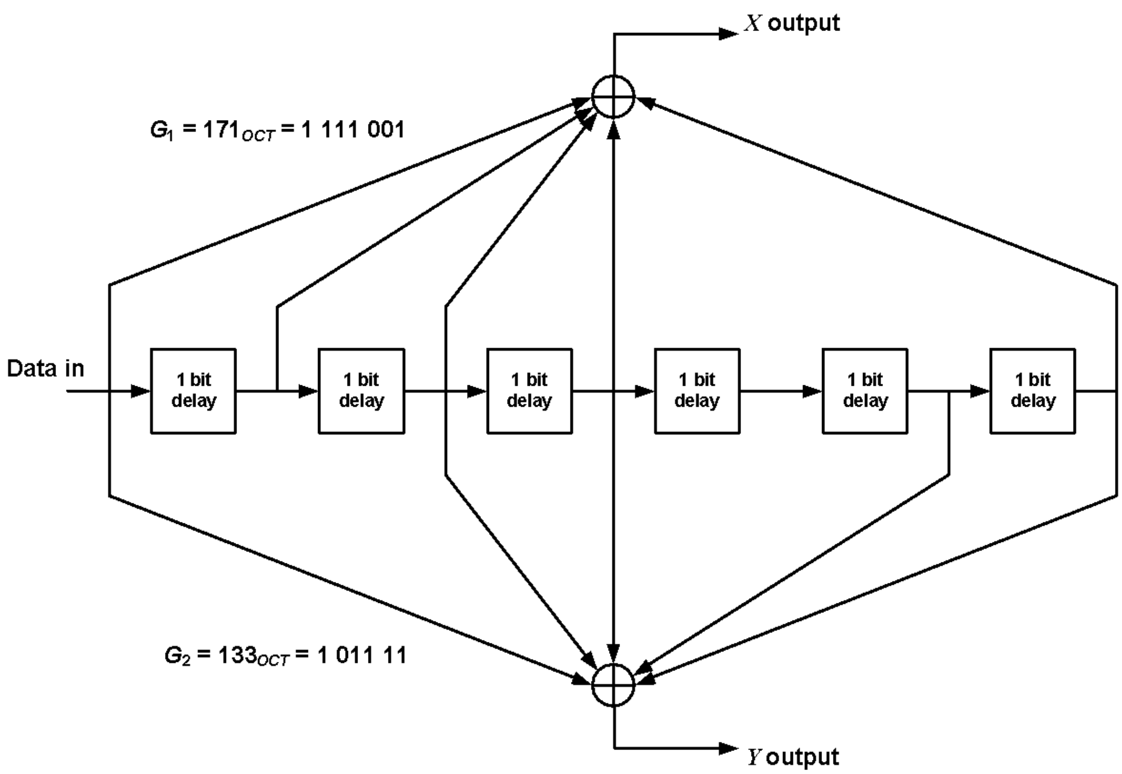


Figure 255—Convolutional encoder of rate 1/2

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.

[4] 8.4.9.2.1

Table 319—The convolutional code with puncturing configuration

	Code Rates		
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.

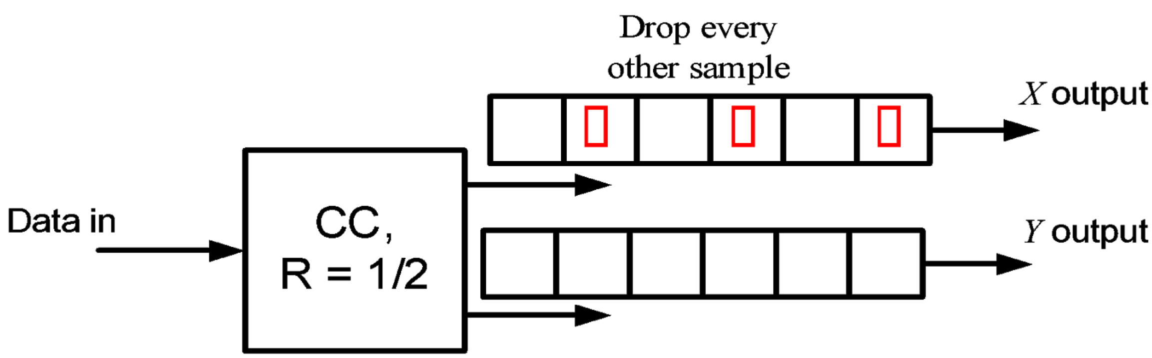


Figure 255b—Convolutional encoder of rate 2/3.
The blocks inside are same as in Figure 255.

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.
    - Then the shift register is initialized as shown in Figure 255c.

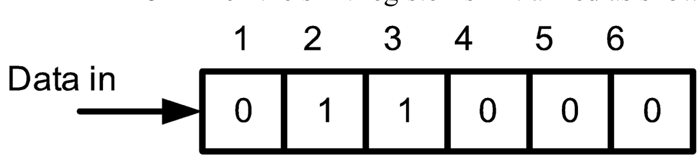


Figure 255c—Initialization example.

- 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

	QPSK		16 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 (bytes)	6						
		9					
	12		12				
	18	18		18	18		
	24		24			24	
		27					27
	30						
	36	36	36	36	36		

- Notice from Table 318:
  - QPSK ½ has a *j* = 6, that is why there are 6 different payloads for this modulation and rate in Table 320.
  - QPSK ¾ has a *j* = 4, that is why there are 4 different payloads for this modulation and rate in Table 320.

- Notice from Table 320:
  - QPSK ½ with 6 bytes data payload will be encoded to 12 bytes and modulated to 12/2 = 6 pairs (1 slot).
  - QPSK ½ with 12 bytes data payload will be encoded to 24 bytes and modulated to 24/2 = 12 pairs (2 slots).
  - QPSK ½ with 36 bytes data payload will be encoded to 72 bytes and modulated to 72/2 = 36 pairs (6 slots).
  - QPSK ¾ with 9 bytes data payload will be encoded to $9\times\frac{4}{3}=12$ bytes and modulated to 6 pairs (1 slot).
  - QPSK ¾ 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
  - A 100 MHz output data has double the data rate and they will be at 100 MHz
  - Use a PLL to generate a 100 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):
    - 55 8A C4 A5 3A 17 24 E1 63 AC 2B F9
- Convolutional Encoded Data (Hex):
    - **0** 28 33 E4 8D 39 20 26 D5 B6 DC 5E 4A F4 7A DD 29 49 4B 6C 89 15 13 48 CA

802.16 e Convolutional Coding, Tail-Biting															
No	Data in HEX	Data in	Shift Register						Y	X	Data out				HEX
			1	2	3	4	5	6							
1		0	1	0	0	1	1	1	0	0	0	0	1	0	2
2	5	1	0	1	0	0	1	1	0	1					
3		0	1	0	1	0	0	1	0	1	1	0	0	0	8
4		1	0	1	0	1	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	1	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	E
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	0	0	0	1	0					
17		1	0	1	0	1	0	0	0	0	0	0	1	1	3
18	C	1	1	0	1	0	1	0	1	1					
19		0	1	1	0	1	0	1	0	1	1	0	0	1	9
20		0	0	1	1	0	1	0	1	0					
21		0	0	0	1	1	0	1	0	0	0	0	1	0	2
22	4	1	0	0	0	1	1	0	0	1					
23		0	1	0	0	0	1	1	0	0	0	0	0	0	0
24		0	0	1	0	0	0	1	0	0					
25		1	0	0	1	0	0	0	0	0	0	0	1	0	2
26	A	0	1	0	0	1	0	0	0	1					
27		1	0	1	0	0	1	0	1	0	0	1	1	0	6

28		0	1 0 1 0 0 1	0 1		
29	5	0	0 1 0 1 0 0	1 1	1 1 0 1	D
30		1	0 0 1 0 1 0	1 0		
31		0	1 0 0 1 0 1	1 0	0 1 0 1	5
32		1	0 1 0 0 1 0	1 0		
33	3	0	1 0 1 0 0 1	0 1	1 0 1 1	B
34		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		
37	A	1	1 1 0 0 1 0	1 1	1 1 0 1	D
38		0	1 1 1 0 0 1	1 0		
39		1	0 1 1 1 0 0	1 1	1 1 0 0	C
40		0	1 0 1 1 1 0	0 0		
41	1	0	0 1 0 1 1 1	1 0	0 1 0 1	5
42		0	0 0 1 0 1 1	1 0		
43		0	0 0 0 1 0 1	1 1	1 1 1 0	E
44		1	0 0 0 0 1 0	0 1		
45	7	0	1 0 0 0 0 1	1 0	0 1 0 0	4
46		1	0 1 0 0 0 0	0 0		
47		1	1 0 1 0 0 0	0 1	1 0 1 0	A
48		1	1 1 0 1 0 0	0 1		
49	2	0	1 1 1 0 1 0	1 1	1 1 1 1	F
50		0	0 1 1 1 0 1	1 1		
51		1	0 0 1 1 1 0	1 0	0 1 0 0	4
52		0	1 0 0 1 1 1	0 0		
53	4	0	0 1 0 0 1 1	1 0	0 1 1 1	7
54		1	0 0 1 0 0 1	1 1		
55		0	1 0 0 1 0 0	0 1	1 0 1 0	A
56		0	0 1 0 0 1 0	0 1		
57	E	1	0 0 1 0 0 1	1 1	1 1 0 1	D
58		1	1 0 0 1 0 0	1 0		
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		
61	1	0	0 1 1 1 0 0	0 0	0 0 1 0	2
62		0	0 0 1 1 1 0	0 1		
63		0	0 0 0 1 1 1	0 1	1 0 0 1	9
64		1	0 0 0 0 1 1	1 0		
65	6	0	1 0 0 0 0 1	1 0	0 1 0 0	4
66		1	0 1 0 0 0 0	0 0		
67		1	1 0 1 0 0 0	0 1	1 0 0 1	9
68		0	1 1 0 1 0 0	1 0		
69	3	0	0 1 1 0 1 0	1 0	0 1 0 0	4
70		0	0 0 1 1 0 1	0 0		
71		1	0 0 0 1 1 0	0 1	1 0 1 1	B
72		1	1 0 0 0 1 1	1 1		
73	A	1	1 1 0 0 0 1	1 0	0 1 1 0	6
74		0	1 1 1 0 0 0	0 1		

75		1	0	1	1	1	0	0	1	1	1 1 0 0	C
76		0	1	0	1	1	1	0	0	0		
77	C	1	0	1	0	1	1	1	0	1	1 0 0 0	8
78		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		
81	2	0	0	0	1	1	0	1	0	0	0 0 0 1	1
82		0	0	0	0	1	1	0	1	0		
83		1	0	0	0	0	1	1	1	0	0 1 0 1	5
84		0	1	0	0	0	0	1	1	0		
85	B	1	0	1	0	0	0	0	0	0	0 0 0 1	1
86		0	1	0	1	0	0	0	1	0		
87		1	0	1	0	1	0	0	0	0	0 0 1 1	3
88		1	1	0	1	0	1	0	1	1		
89	F	1	1	1	0	1	0	1	1	0	0 1 0 0	4
90		1	1	1	1	0	1	0	0	0		
91		1	1	1	1	1	0	1	0	1	1 0 0 0	8
92		1	1	1	1	1	1	0	0	0		
93	9	1	1	1	1	1	1	1	1	1	1 1 0 0	C
94		0	1	1	1	1	1	1	0	0		
95		0	0	1	1	1	1	1	0	1	1 0 1 0	A
96		1	0	0	1	1	1	1	0	1		

5.

Interleaving[5]

- 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.
  - 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.
  - That is: 2, 4, or 6 for QPSK, 16-QAM or 64-QAM, respectively.
- Let $s = N_{cpc}/2$.
- Within a block of N_{cbps} bits at transmission, let:
  - k be the index of the coded bit before the first permutation,
  - m_k be the index of that coded bit after the first and before the second permutation,
  - j_k be the index after the second permutation, just prior to modulation mapping, and
  - d be the modulo used for the permutation.

- The first permutation is defined by Equation (130):

$$m_k = \frac{N_{\text{cbps}}}{d} \cdot k_{\text{mod}(d)} + \left\lfloor \frac{k}{d} \right\rfloor; \qquad k = 0,1,\ldots,N_{\text{cbps}} - 1, \qquad d = 16 \tag{130}$$

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

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

(131)

A. RTL Implementation

- Refer to Example 1.
  - N_{cbps} = 192 bits, N_{cpc} = 2 bits, s = 1.
  - $$m_k = \frac{192}{16} \cdot k_{\text{mod}(16)} + \left\lfloor \frac{k}{16} \right\rfloor = 12 \cdot k_{\text{mod}(16)} + \left\lfloor \frac{k}{16} \right\rfloor; \qquad k = 0,1,\ldots,191.$$
  - $$j_k = \lfloor m_k \rfloor + \left(m_k + 192 - \left\lfloor \frac{16 \cdot m_k}{192} \right\rfloor \right)_{\text{mod}(1)} = m_k; \qquad k = 0,1,\ldots,191.$$
- Notice that here $j_k = m_k$. That is only one permutation.

[5] 8.4.9.3

B. Testbench Gold data

802.16 e

Bit Interleaving, s = 1

No	Data in HEX	Data in DEC	Data in	kmod16	mk		mk	Data out	HEX
0	2	2	0	0	0		0	0	4
1			0	1	12		1	1	
2			1	2	24		2	0	
3			0	3	36		3	0	
4	8	8	1	4	48		4	1	B
5			0	5	60		5	0	
6			0	6	72		6	1	
7			0	7	84		7	1	
8	3	3	0	8	96		8	0	0
9			0	9	108		9	0	
10			1	10	120		10	0	
11			1	11	132		11	0	
12	3	3	0	12	144		12	0	4
13			0	13	156		13	1	
14			1	14	168		14	0	
15			1	15	180		15	0	
16	E	14	1	0	1		16	0	7
17			1	1	13		17	1	
18			1	2	25		18	1	
19			0	3	37		19	1	
20	4	4	0	4	49		20	1	D
21			1	5	61		21	1	
22			0	6	73		22	0	
23			0	7	85		23	1	
24	8	8	1	8	97		24	1	F
25			0	9	109		25	1	
26			0	10	121		26	1	
27			0	11	133		27	1	
28	D	13	1	12	145		28	1	A
29			1	13	157		29	0	
30			0	14	169		30	1	
31			1	15	181		31	0	
32	3	3	0	0	2		32	0	4
33			0	1	14		33	1	
34			1	2	26		34	0	

35			1	3	38
36	9	9	1	4	50
37			0	5	62
38			0	6	74
39			1	7	86
40	2	2	0	8	98
41			0	9	110
42			1	10	122
43			0	11	134
44	0	0	0	12	146
45			0	13	158
46			0	14	170
47			0	15	182
48	2	2	0	0	3
49			0	1	15
50			1	2	27
51			0	3	39
52	6	6	0	4	51
53			1	5	63
54			1	6	75
55			0	7	87
56	D	13	1	8	99
57			1	9	111
58			0	10	123
59			1	11	135
60	5	5	0	12	147
61			1	13	159
62			0	14	171
63			1	15	183
64	B	11	1	0	4
65			0	1	16
66			1	2	28
67			1	3	40
68	6	6	0	4	52
69			1	5	64
70			1	6	76
71			0	7	88
72	D	13	1	8	100
73			1	9	112
74			0	10	124
75			1	11	136
76	C	12	1	12	148
77			1	13	160
78			0	14	172
79			0	15	184
80			0	0	5
81	5	5	1	1	17

35	0	
36	0	2
37	0	
38	1	
39	0	
40	1	F
41	1	
42	1	
43	1	
44	0	2
45	0	
46	1	
47	0	
48	1	A
49	0	
50	1	
51	0	
52	0	5
53	1	
54	0	
55	1	
56	1	D
57	1	
58	0	
59	1	
60	0	5
61	1	
62	0	
63	1	
64	1	F
65	1	
66	1	
67	1	
68	0	6
69	1	
70	1	
71	0	
72	0	1
73	0	
74	0	
75	1	
76	1	C
77	1	
78	0	
79	0	
80	0	
81	0	0

82			0	2	29		82	0	
83			1	3	41		83	0	
84	E	14	1	4	53		84	0	2
85			1	5	65		85	0	
86			1	6	77		86	1	
87			0	7	89		87	0	
88	4	4	0	8	101		88	0	1
89			1	9	113		89	0	
90			0	10	125		90	0	
91			0	11	137		91	1	
92	A	10	1	12	149		92	1	A
93			0	13	161		93	0	
94			1	14	173		94	1	
95			0	15	185		95	0	
96	F	15	1	0	6		96	0	5
97			1	1	18		97	1	
98			1	2	30		98	0	
99			1	3	42		99	1	
100	4	4	0	4	54		100	1	8
101			1	5	66		101	0	
102			0	6	78		102	0	
103			0	7	90		103	0	
104	7	7	0	8	102		104	0	5
105			1	9	114		105	1	
106			1	10	126		106	0	
107			1	11	138		107	1	
108	A	10	1	12	150		108	0	1
109			0	13	162		109	0	
110			1	14	174		110	0	
111			0	15	186		111	1	
112	D	13	1	0	7		112	1	E
113			1	1	19		113	1	

114			0	2	31	114	1	
115			1	3	43	115	0	
116	D	13	1	4	55	116	1	9
117			1	5	67	117	0	
118			0	6	79	118	0	
119			1	7	91	119	1	
120	2	2	0	8	103	120	1	A
121			0	9	115	121	0	
122			1	10	127	122	1	
123			0	11	139	123	0	
124	9	9	1	12	151	124	0	3
125			0	13	163	125	0	
126			0	14	175	126	1	
127			1	15	187	127	1	
128	4	4	0	0	8	128	0	0
129			1	1	20	129	0	
130			0	2	32	130	0	
131			0	3	44	131	0	
132	9	9	1	4	56	132	1	9
133			0	5	68	133	0	
134			0	6	80	134	0	
135			1	7	92	135	1	
136	4	4	0	8	104	136	1	A

137			1	9	116	137	0	
138			0	10	128	138	1	
139			0	11	140	139	0	
140	B	11	1	12	152	140	0	2
141			0	13	164	141	0	
142			1	14	176	142	1	
143			1	15	188	143	0	
144	6	6	0	0	9	144	0	4
145			1	1	21	145	1	
146			1	2	33	146	0	
147			0	3	45	147	0	
148	C	12	1	4	57	148	1	F
149			1	5	69	149	1	
150			0	6	81	150	1	
151			0	7	93	151	1	
152	8	8	1	8	105	152	1	D
153			0	9	117	153	1	
154			0	10	129	154	0	
155			0	11	141	155	1	
156	9	9	1	12	153	156	0	5
157			0	13	165	157	1	
158			0	14	177	158	0	
159			1	15	189	159	1	

160	1	1	0	0	10
161			0	1	22
162			0	2	34
163			1	3	46
164	5	5	0	4	58
165			1	5	70
166			0	6	82
167			1	7	94
168	1	1	0	8	106
169			0	9	118
170			0	10	130
171			1	11	142
172	3	3	0	12	154
173			0	13	166
174			1	14	178
175			1	15	190
176	4	4	0	0	11
177			1	1	23
178			0	2	35
179			0	3	47
180	8	8	1	4	59
181			0	5	71
182			0	6	83

160	1	8
161	0	
162	0	
163	0	
164	0	0
165	0	
166	0	
167	0	
168	1	8
169	0	
170	0	
171	0	
172	0	6
173	1	
174	1	
175	0	
176	1	B
177	0	
178	1	
179	1	
180	1	D
181	1	
182	0	

183			0	7	95		183	1	
184	C	12	1	8	107		184	0	1
185			1	9	119		185	0	
186			0	10	131		186	0	
187			0	11	143		187	1	
188	A	10	1	12	155		188	1	E
189			0	13	167		189	1	
190			1	14	179		190	1	
191			0	15	191		191	0	

6.

Modulation[6]

A. Data modulation[7]

- 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.

$$P_{av} = \tfrac{1}{4}\left[4\left(1^2 + 1^2\right)\right] = 2 \quad \Rightarrow \quad c = 1/\sqrt{P_{av}} = 1/\sqrt{2}$$

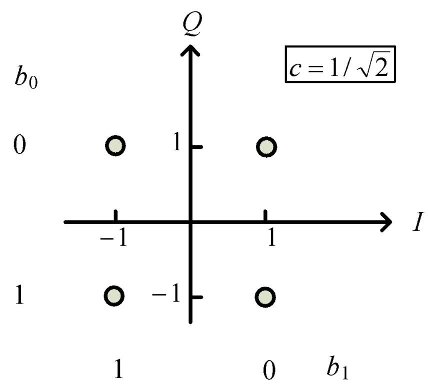


Figure 263a—QPSK constellations

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 Constellation Mapping					
No	Int. Data in	Constallation Mapping		Normalized Constellation Mapping	
0 1 2 3	0 1 0 0	1 1	-1 1	0.707 0.707	-0.707 0.707
4 5 6 7	1 0 1 1	-1 -1	1 -1	-0.707 -0.707	0.707 -0.707
8 9	0 0	1	1	0.707	0.707

[6] 8.4.9.4
[7] 8.4.9.4.2

10 11	0 0	1	1	0.707	0.707
12 13	0 1	1	-1	0.707	-0.707
14 15	0 0	1	1	0.707	0.707
16 17	0 1	1	-1	0.707	-0.707
18 19	1 1	-1	-1	-0.707	-0.707
20 21	1 1	-1	-1	-0.707	-0.707
22 23	0 1	1	-1	0.707	-0.707
24 25	1 1	-1	-1	-0.707	-0.707
26 27	1 1	-1	-1	-0.707	-0.707
28 29	1 0	-1	1	-0.707	0.707
30 31	1 0	-1	1	-0.707	0.707
32 33	0 1	1	-1	0.707	-0.707
34 35	0 0	1	1	0.707	0.707
36 37	0 0	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 51	1 0	-1	1	-0.707	0.707
52 53	0 1	1	-1	0.707	-0.707
54 55	0 1	1	-1	0.707	-0.707
56	1	-1	-1	-0.707	-0.707

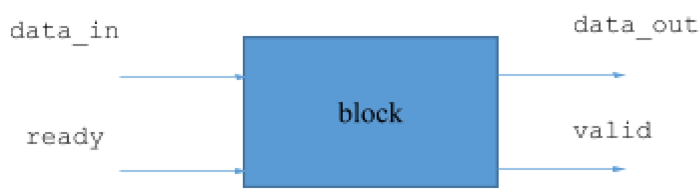
57 58 59	1 0 1	1	-1	0.707	-0.707
60 61	0 1	1	-1	0.707	-0.707
62 63	0 1	1	-1	0.707	-0.707
64 65	1 1	-1	-1	-0.707	-0.707
66 67	1 1	-1	-1	-0.707	-0.707
68 69	0 1	1	-1	0.707	-0.707
70 71	1 0	-1	1	-0.707	0.707
72 73	0 0	1	1	0.707	0.707
74 75	0 1	1	-1	0.707	-0.707
76 77	1 1	-1	-1	-0.707	-0.707
78 79	0 0	1	1	0.707	0.707
80 81	0 0	1	1	0.707	0.707
82 83	0 0	1	1	0.707	0.707
84 85	0 0	1	1	0.707	0.707
86 87	1 0	-1	1	-0.707	0.707
88 89	0 0	1	1	0.707	0.707
90 91	0 1	1	-1	0.707	-0.707
92 93	1 0	-1	1	-0.707	0.707
94 95	1 0	-1	1	-0.707	0.707
96 97	0 1	1	-1	0.707	-0.707
98 99	0 1	1	-1	0.707	-0.707
100 101	1 0	-1	1	-0.707	0.707
102 103	0 0	1	1	0.707	0.707

104 105	0 1	1	-1	0.707	-0.707
106 107	0 1	1	-1	0.707	-0.707
108 109	0 0	1	1	0.707	0.707
110 111	0 1	1	-1	0.707	-0.707
112 113	1 1	-1	-1	-0.707	-0.707
114 115	1 0	-1	1	-0.707	0.707
116 117	1 0	-1	1	-0.707	0.707
118 119	0 1	1	-1	0.707	-0.707
120 121	1 0	-1	1	-0.707	0.707
122 123	1 0	-1	1	-0.707	0.707
124 125	0 0	1	1	0.707	0.707
126 127	1 1	-1	-1	-0.707	-0.707
128 129	0 0	1	1	0.707	0.707
130 131	0 0	1	1	0.707	0.707
132 133	1 0	-1	1	-0.707	0.707
134 135	0 1	1	-1	0.707	-0.707
136 137	1 0	-1	1	-0.707	0.707
138 139	1 0	-1	1	-0.707	0.707
140 141	0 0	1	1	0.707	0.707
142 143	1 0	-1	1	-0.707	0.707
144 145	0 1	1	-1	0.707	-0.707
146 147	0 0	1	1	0.707	0.707
148 149	1 1	-1	-1	-0.707	-0.707
150	1	-1	-1	-0.707	-0.707

151	1				
152	1	-1	-1	-0.707	-0.707
153	1				
154	0	1	-1	0.707	-0.707
155	1				
156	0	1	-1	0.707	-0.707
157	1				
158	0	1	-1	0.707	-0.707
159	1				
160	1	-1	1	-0.707	0.707
161	0				
162	0	1	1	0.707	0.707
163	0				
164	0	1	1	0.707	0.707
165	0				
166	0	1	1	0.707	0.707
167	0				
168	1	-1	1	-0.707	0.707
169	0				
170	0	1	1	0.707	0.707
171	0				
172	0	1	-1	0.707	-0.707
173	1				
174	1	-1	1	-0.707	0.707
175	0				
176	1	-1	1	-0.707	0.707
177	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. 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. Each block has an input data line (`data_in`) and input data ready line (`ready`). The ready line is asserted when there is a valid data presented on the input data. Each block has an output data line (`data_out`) and a valid output signal (`valid`). The block will assert the valid output signal when there is a valid data at the output of the 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.

8. 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