Convolutional Encoder and Viterbi Decoder Design for WARP OFDM PHY V15.0

Yang Sun (ysun@rice.edu)
2010-Oct-16

Summary:

This document summarizes the hardware design specs for a convolutional encoder and a Viterbi decoder for WARP OFDM PHY v15.0 (svn rev 1580 for FPGA v1 and svn rev 1585 for FPGA v2). The design is built using the 10.1 release of the Xilinx tools (ISE 10.1.03 + IP3, Sysgen 10.1.3.1386).

Code Structure:

In this design, a K=7 convolutional code is used. The code structure and the puncture pattern are compliant with IEEE 802.11a standard. The following two figures are copied from the standard. As shown in Fig. 114, the base coding rate is 1/2. Higher code rates (R=2/3 and R=3/4) are obtained through puncture (cf. Fig. 115).

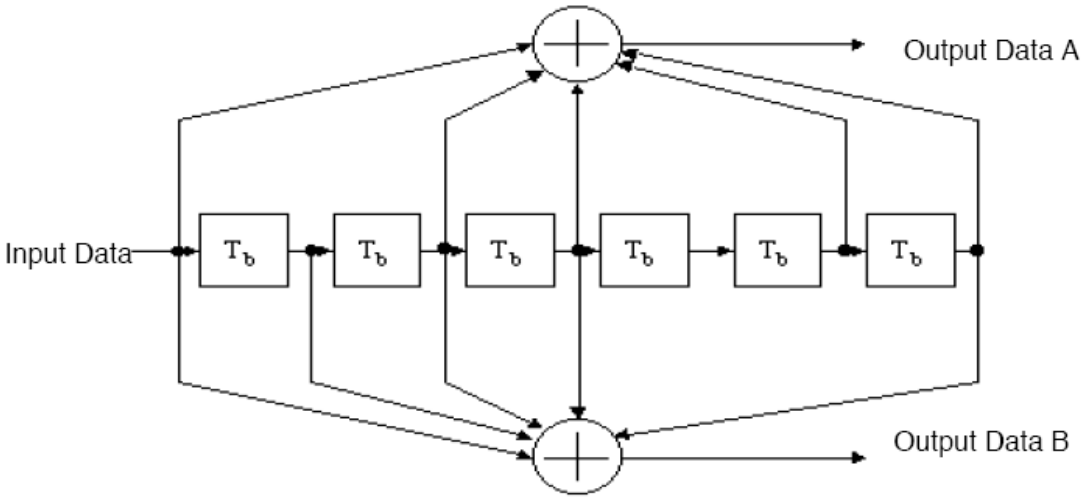


Figure 114—Convolutional encoder (k = 7)

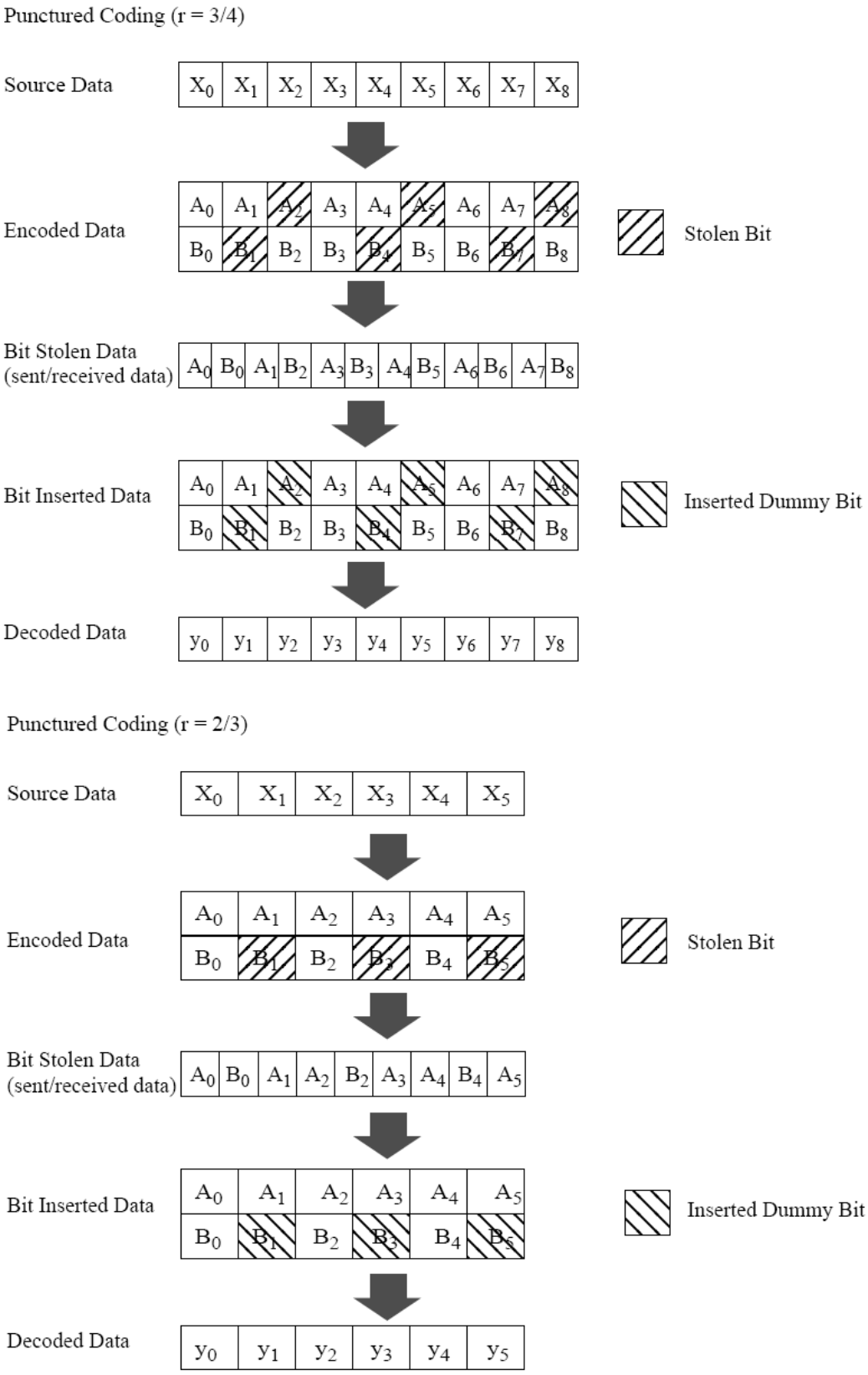


Figure 115—An example of the bit-stealing and bit-insertion procedure (r = 3/4, 2/3)

Description of the FEC Codec:

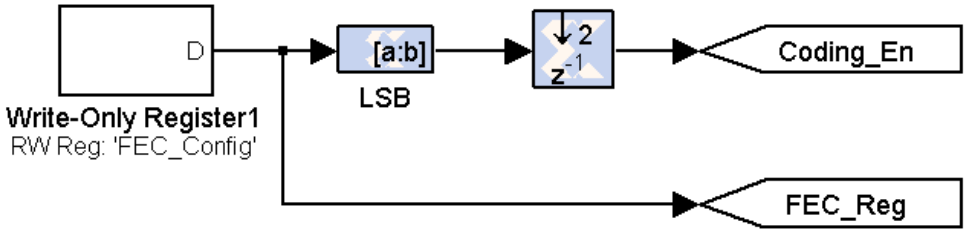
The FEC codec supports all three modes of the currently WARP OFDM PHY: 1) SISO mode, 2) 2x2 MIMO mode, and 3) Alamouti mode. The FEC codec supports three modulation types: 1) BPSK, 2) QPSK, and 3) 16-QAM. 64-QAM is not supported at this time. The coding can be turned on and off by programming the "FEC_Config" register. The coding rate can be changed by modifying the second byte of the packet header.

Configuration Register

A 32-bit FEC configuration register (FEC_Config) is created in block TxRx Registers. The definition of the FEC_Config register is as follows.

Bits	Name	Functionality	Notes
0	Coding_enable	'0': Coding is turned off. The entire packet will be uncoded. '1': Coding is turned on. The header is always rate 1/2 coded. The coding rate for the data payload can be 1/2, 2/3, 3/4, or 1.	Coding rate of 1 means no coding.
1	Soft_decoding	'0': The demapper will produce hard decision values for the Viterbi decoder. '1': The demapper will produce 4-bit soft LLR values for the Viterbi decoder.	Theoretically, soft decoding will give a better performance. The performance needs to be verified on hardware.
2	Zero_tail	'0': The code is not zero terminated. The decoder will not use zero state as the initial state to do the trace back for the last section of the trellis. '1': The code is zero terminated. The decoder will use zero state as the initial state to do the trace back for the last section of the trellis.	If the convolutional code is not zero terminated. This bit MUST be set to '0'. Otherwise, the decoding will always fail. For a non zero-tailing code, we suggest to leave the last 2-4 bytes of the header and the data payload unused. If the convolutional code is zero-terminated, this bit should be set to '1'. However, it is still OK to set it to '0' (may loose some performance).
3	Not used		
4-7	Scaling_qpsk	The scaling factor for the soft demapper for the QPSK signal.	The soft demapper will use this factor to scale the LLR

			value to a 4-bit fixed-point value. A recommend value is 6. This needs to be tested in HW to find an optimal scaling factor.
8-12	Scaling_16qam	The scaling factor for the soft demapper for the 16-QAM signal.	The soft demapper will use this factor to scale the LLR value to 4-bit fixed-point value. A recommend value is 16. This needs to be tested in HW to find an optimal scaling factor.
13-31	Not used		



Header Control Bits

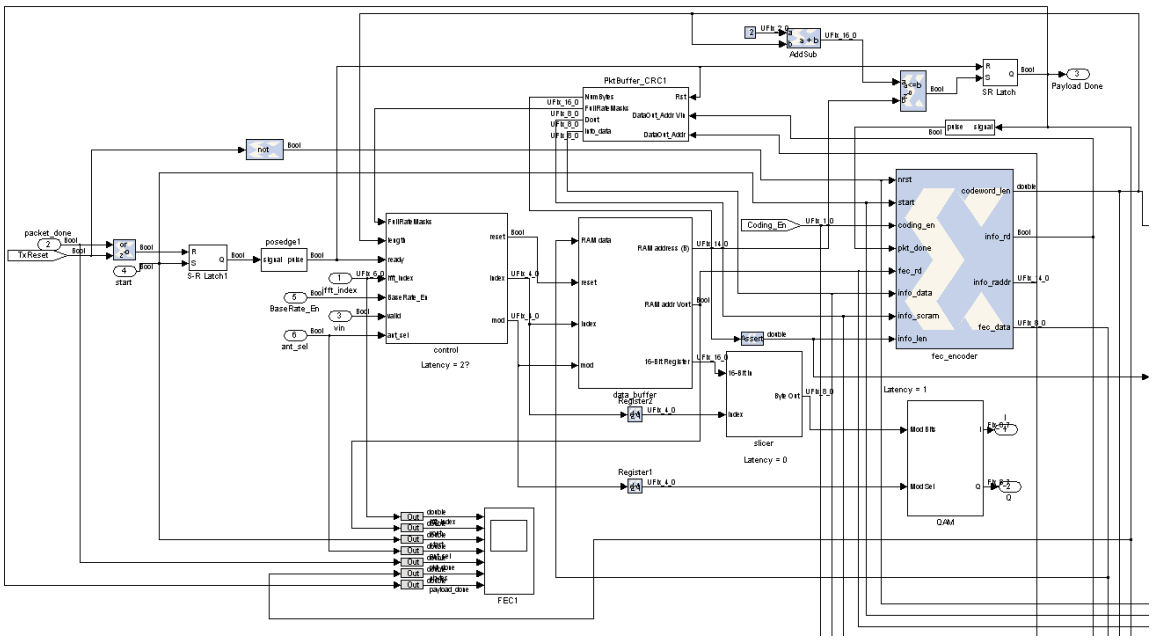
The second byte of the header controls the coding rate for the data payload. If the coding is turned on, the encoder will look for this value to set the correct coding rate. The decoder will also need to know this value to de-puncture the bit stream.

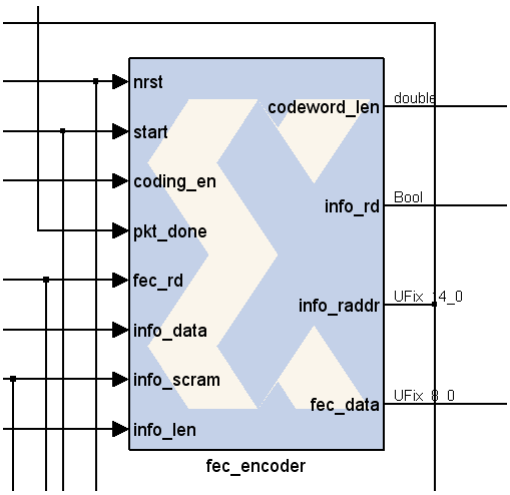
Value	Coding Rate for Data Payload
0	1/2
1	2/3
2	3/4
3	1 (No coding)

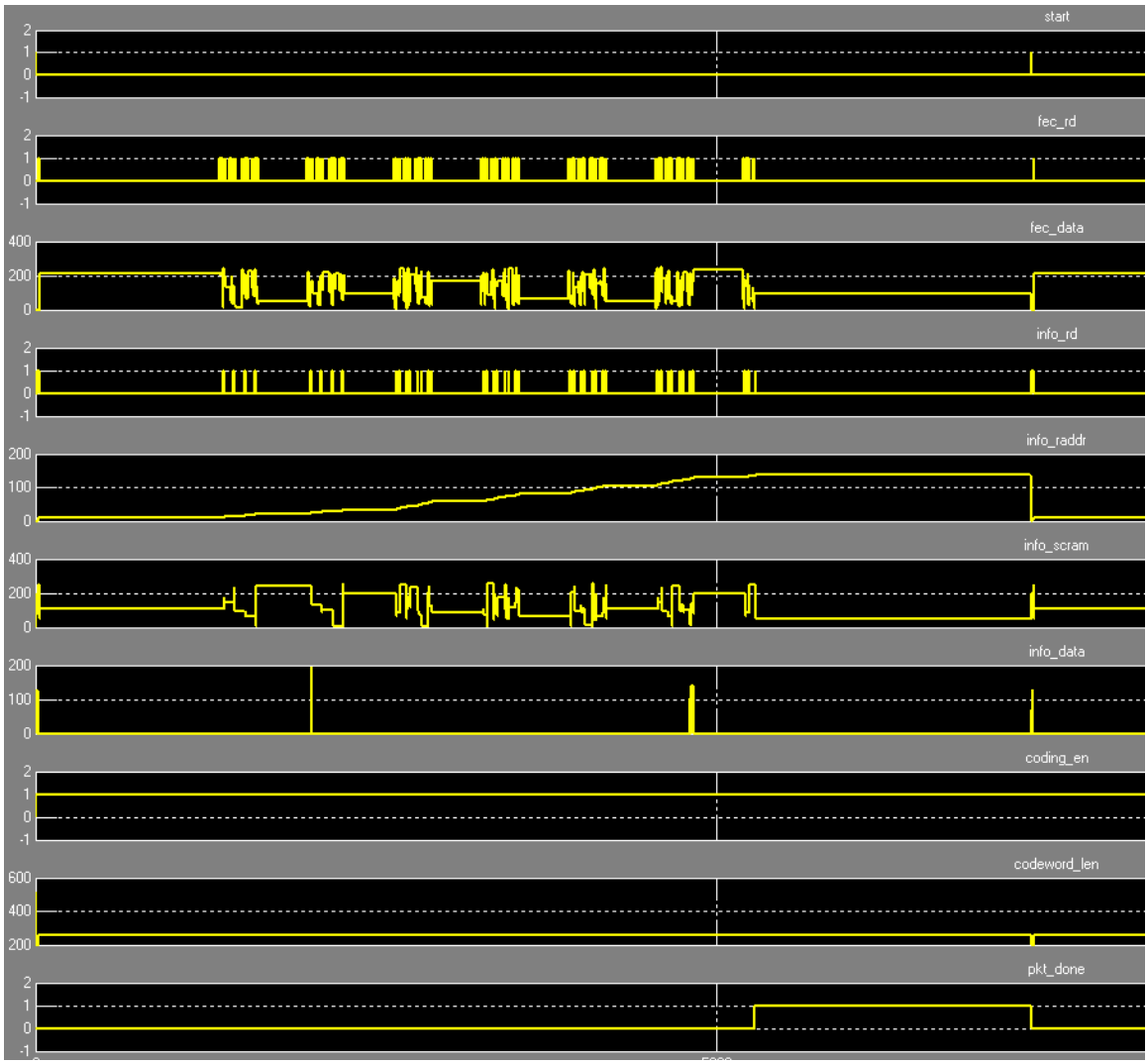
FEC Encoder Implementation:

The convolutional encoder is implemented with Verilog and is integrated into the sysgen model as a block-box. The following figure shows the connection between the encoder and the rest of the sysgen blocks. As can be seen, the encoder sits between the "data_buffer" block and the "PktBuffer_CRC1" block. The encoder will pre-fetch the data (scrambled information data) from the "PktBuffer_CRC1" block and encoded it. The encoded bits are stored into a local small buffer. When this buffer is full, the encoder will stop fetching data the "PktBuffer_CRC1" block. When the encoder sees a new data byte request from the "data_buffer" block, it will return a coded data byte to the "data_buffer" block. When the coding is turned off, i.e. coding_en = 0, the encoder will bypass the scrambled information data to the "data_buffer". The encoder needs to calculate the actual number of bytes for transmission. A multiplier is used to compute the codeword length. Let N be the original number of bytes for transmission (header + payload + CRC). The following table summarizes the codeword length. Note that when coding is enabled, the number of the base rate symbols needs to be doubled. In the hardware implementation, the codeword length is computed approximated, which might be 1-2 bytes larger than the exact value. (PS. I wasn't sure why "2" is added to num of bytes in the original sysgen design).

Coding Enable	Coding Rate	Codeword Length (bytes)
No	-	N
Yes	1/2	2N
Yes	2/3	48 + (N-24) * 4/3
Yes	3/4	48 + (N-24) * 3/2
Yes	1	N + 24



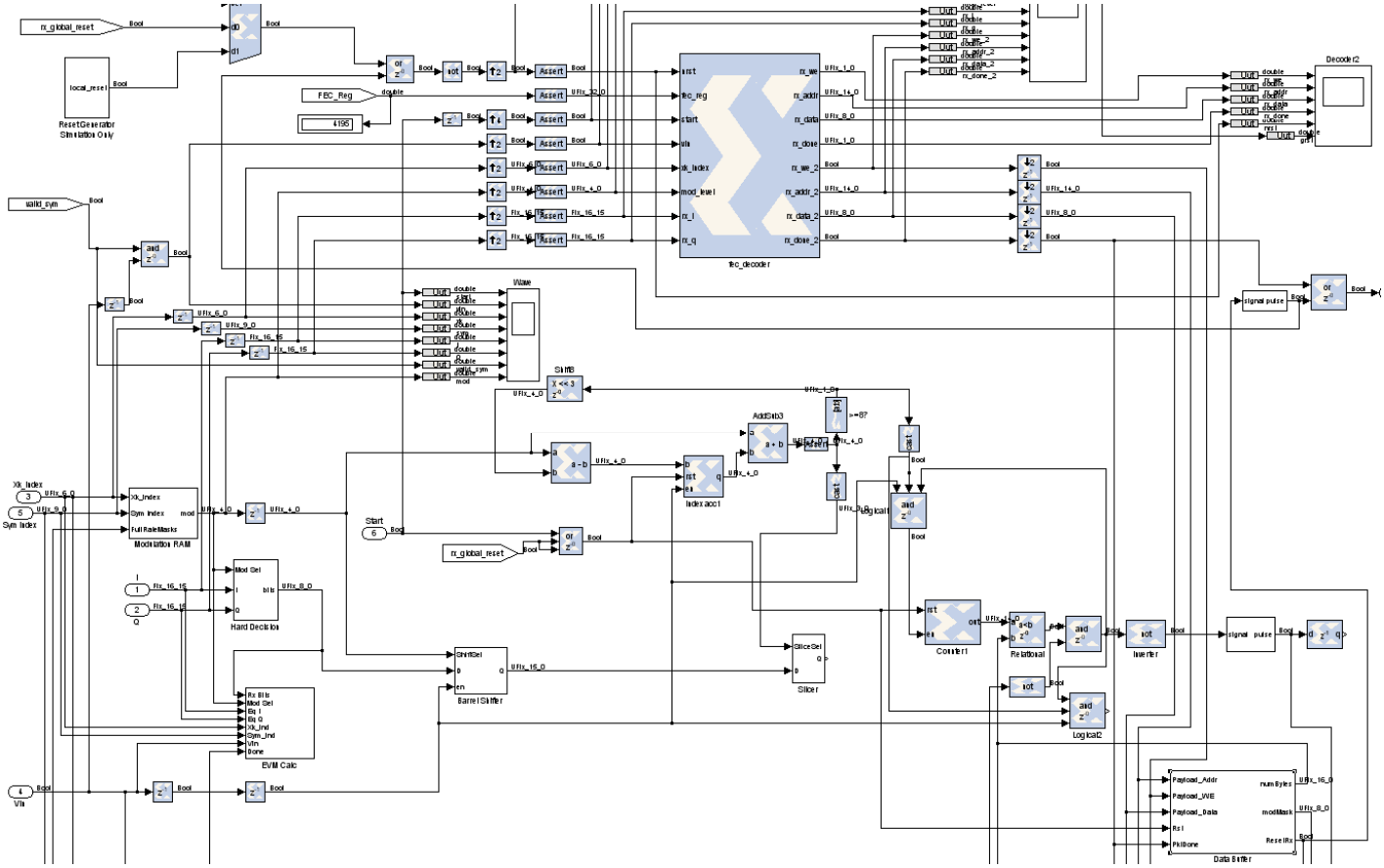


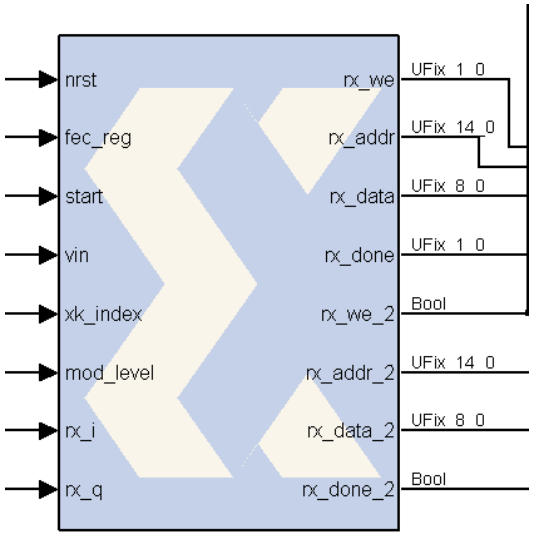


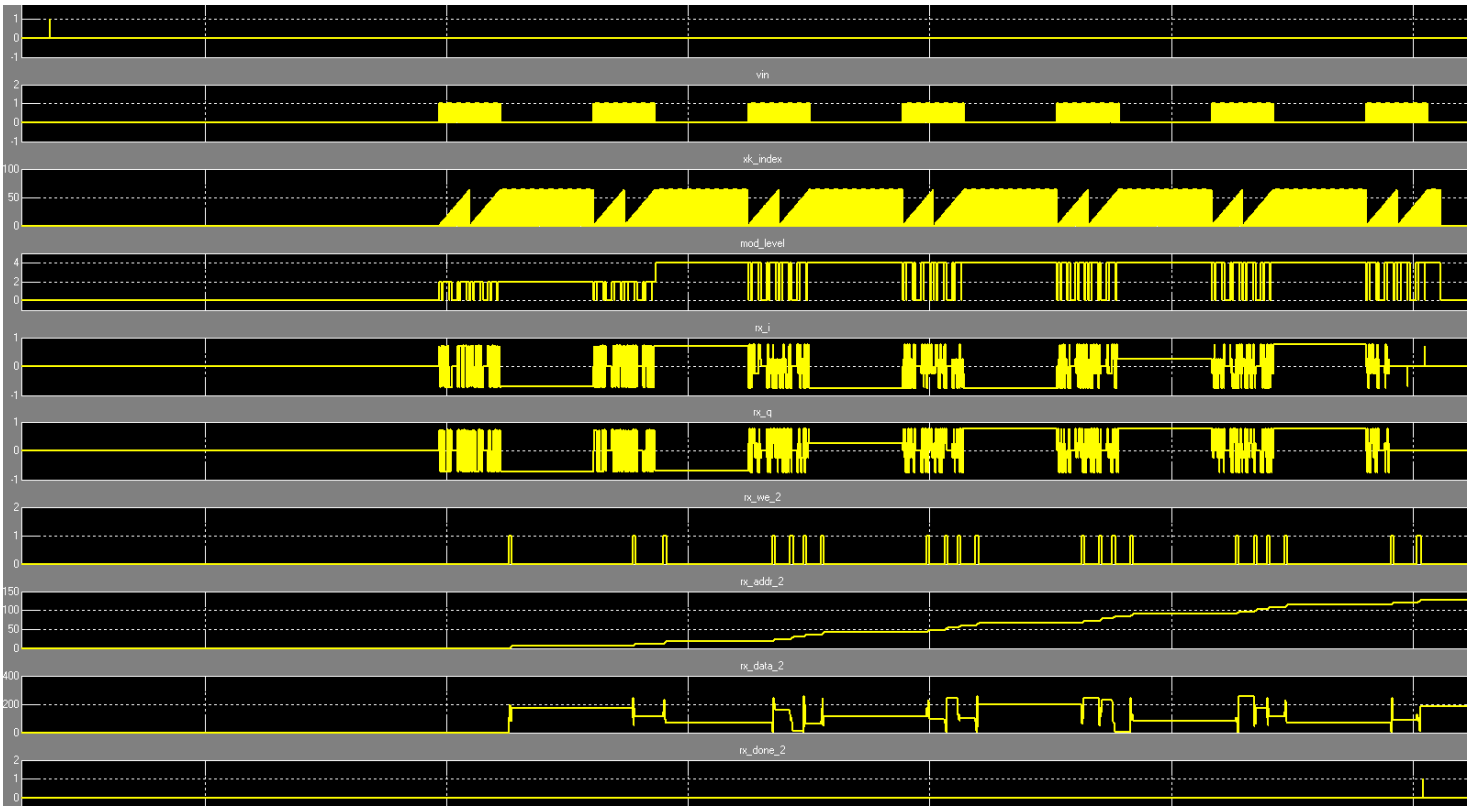
Alamouti simulation

FEC Decoder Implementation:

The FEC decoder is also implemented with Verilog and is integrated into sysgen as a black-box. The following figure shows the connection between the FEC decoder and the other sysgen blocks. The FEC decoder takes I and Q data and produce the decoded data in bytes. The decoded data are then sent to the "Data Buffer" block for further processing, e.g. error checking.





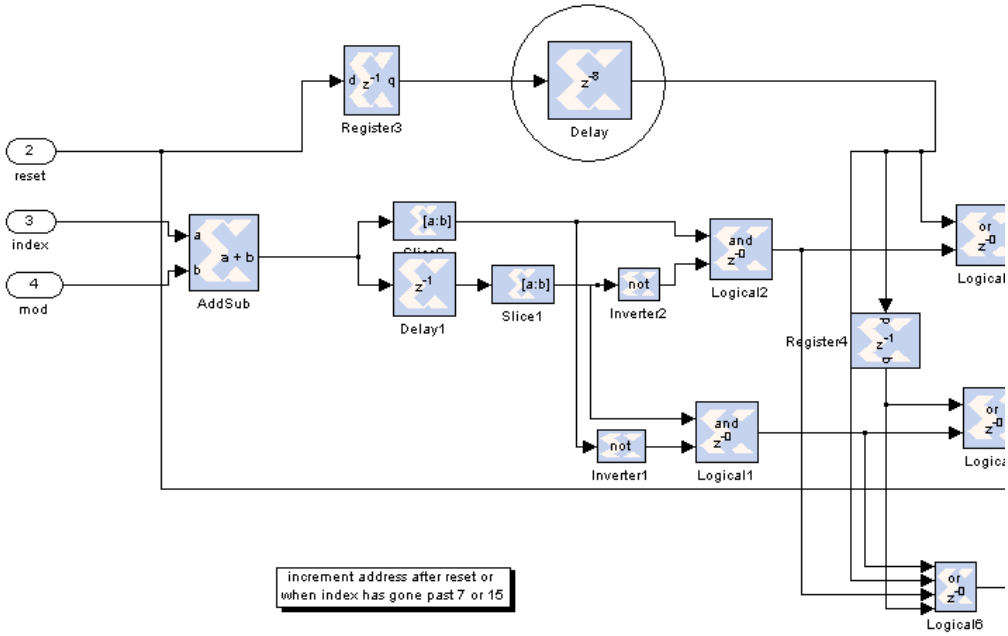


Alamouti simulation

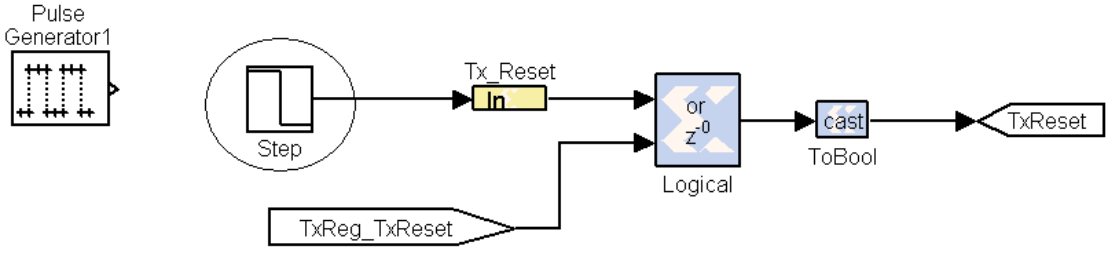
Sysgen Model Modification:

Other than the "FlexibleMod" block and the "Packet_Constructor" block, the sysgen model was modified at several other places to support coding.

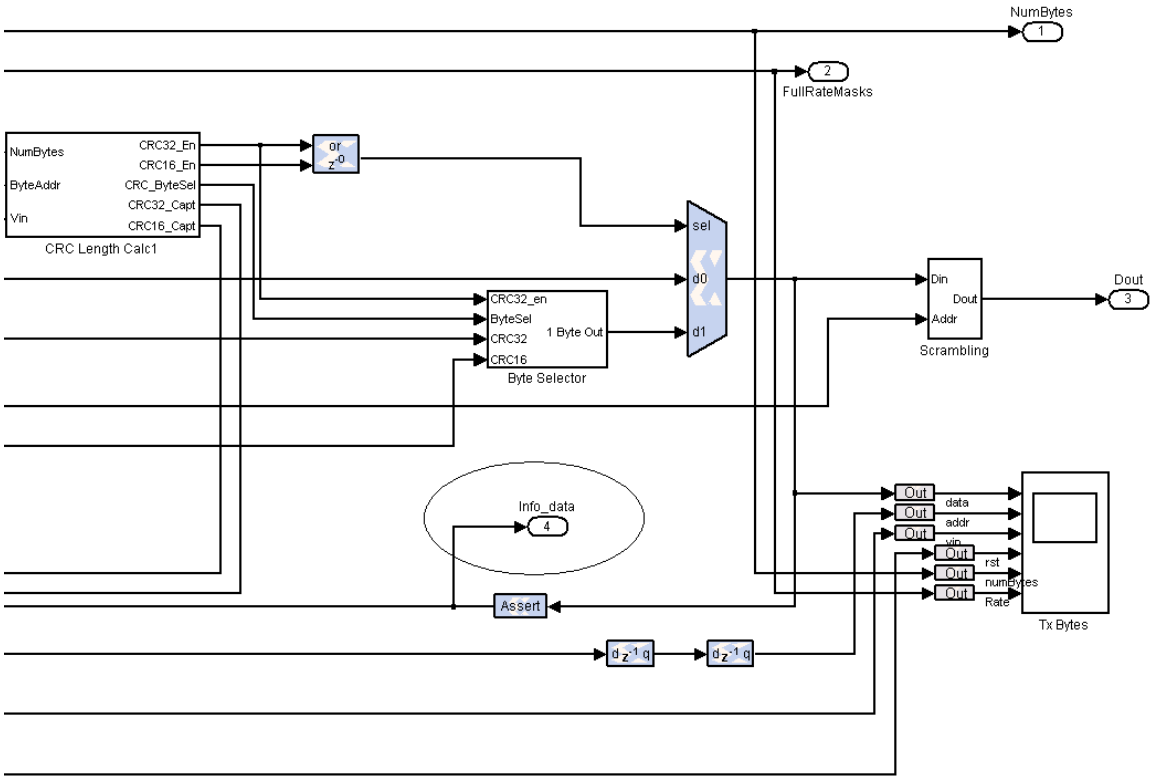
A change was made to the "data_buffer" block in the transmitter. A 8 cycle delay register was added to delay the reset signal so that the FEC encoder will have enough time to pre-fetch the data from the "PktBuffer_CRC1" block for encoding.



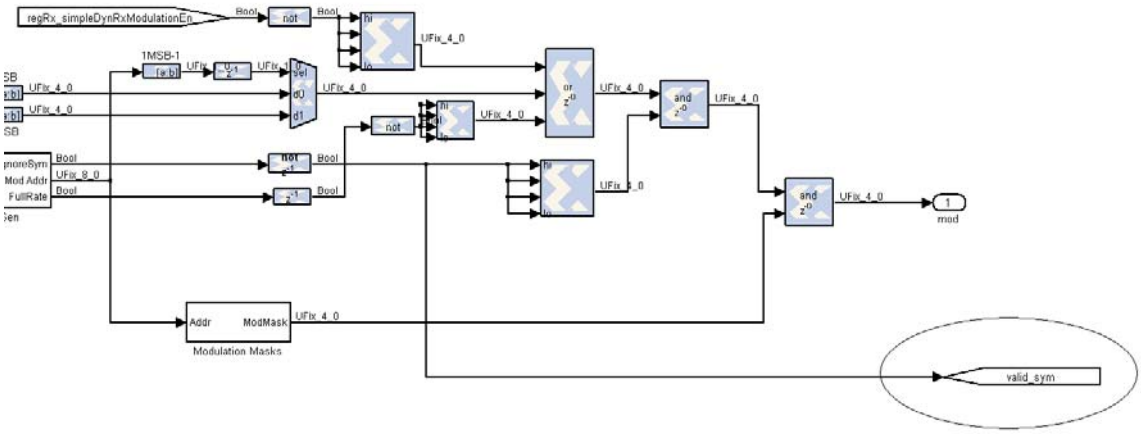
A change was made to the "Tx Reset Logic" block. A step function is applied to the Tx_Reset port to provide a reset signal to the encoder.



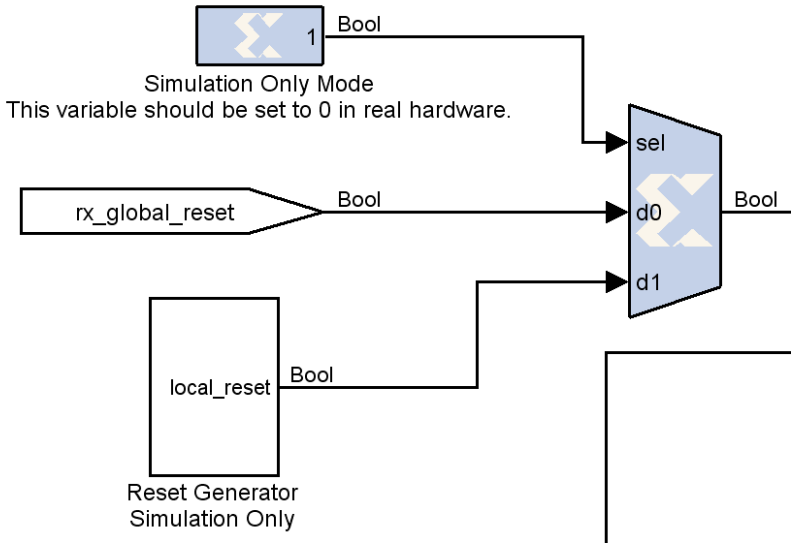
A change was made to the "PktBuffer_CRC1" block in the transmitter. A new output port "info_data" was created. This data is the un-scrambled data or information data. The encoder uses this signal to get the value of the second byte of the header. This value controls the coding rate for the data payload.



A change was made to the "Packet_Constructor/Modulation RAM" block. A tag "valid_sym" was created from an existing signal. This signal is gated with the "vin" signal to filter the unwanted "vin" signal.



A local reset signal was generated in the "Packet_Constructor" block to reset the FEC decoder at time 0. Because applying a step function to the Rx_Reset port will break the sysgen simulation, a local reset generation block is needed for simulation. Note that when generating hardware, the selector of the mux block should be set to '0'.



Matlab Script Modification:

The following lines were added to the "ofdm_tx_supermimo_init.m"

```
fec_coding_en = 1 ;
fec_soft_dec = 1 ;
fec_zero_tail = 0 ;
fec_qpsk_scl = 6 ;
fec_16qam_scl = 16 ;
fec_code_rate = 0 ;  % valid values are [0, 1, 2, 3] meaning rate 1/2, rate 2/3, rate 3/4, and rate 1 (no coding)
FEC_Config = fec_coding_en*1 + fec_soft_dec*2 + fec_qpsk_scl*2^4 + fec_16qam_scl*2^8 ;

% Header is always 1/2 coded, double the base rate symbol if coding is enabled.
if(fec_coding_en)
   numBaseRateSymbols = numBaseRateSymbols *2 ;
end
```

Hardware Resource Estimation:

The FEC codec will take about 12% of the slices in Virtex2-Pro FPGA. Three multipliers are used.