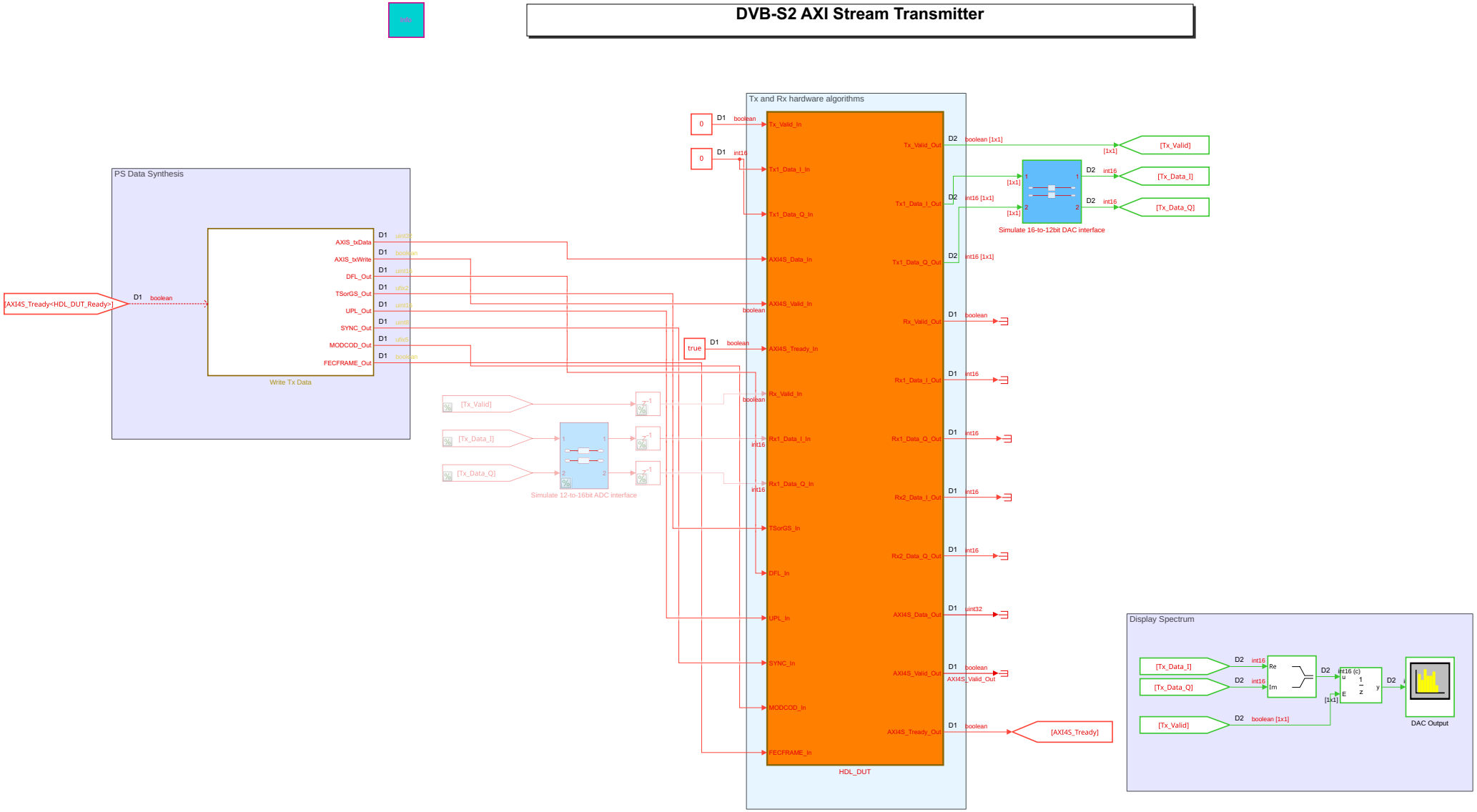
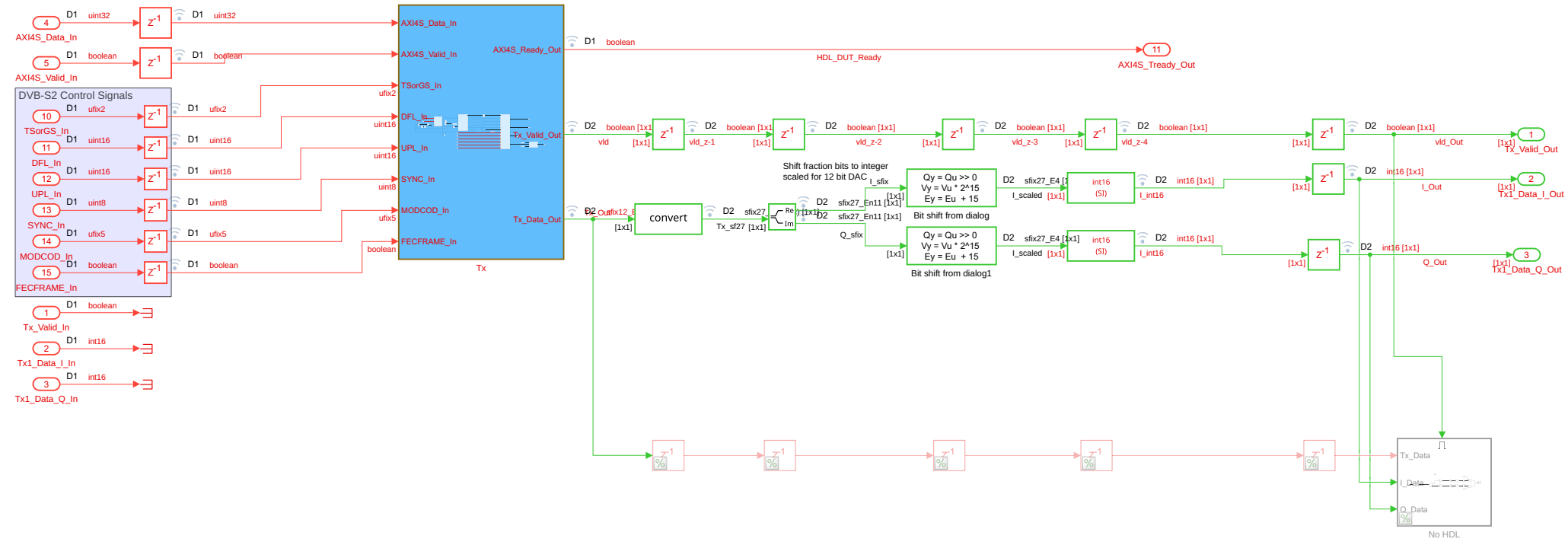
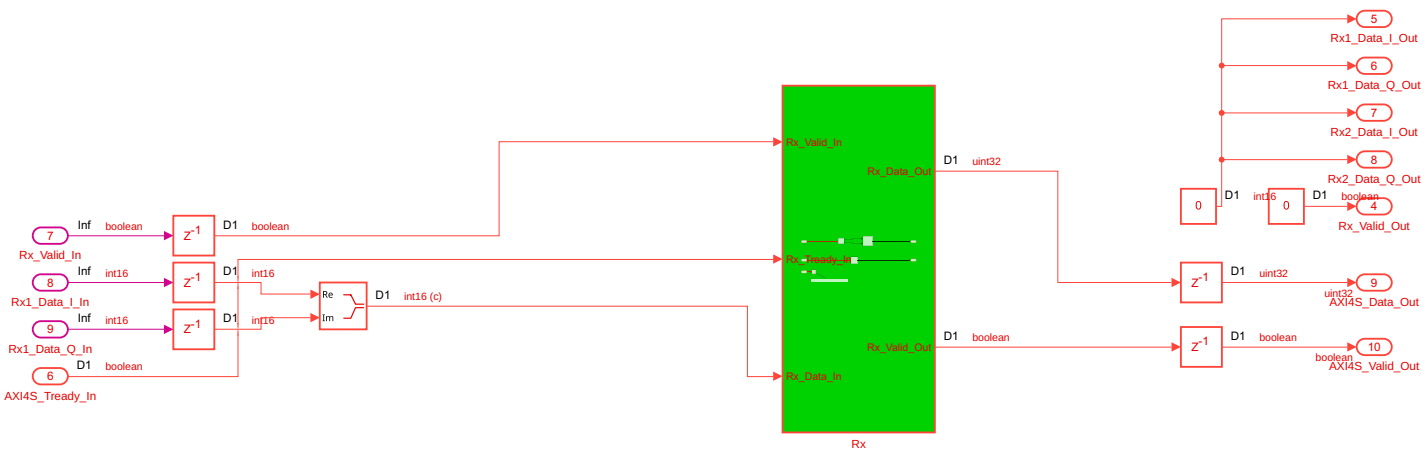
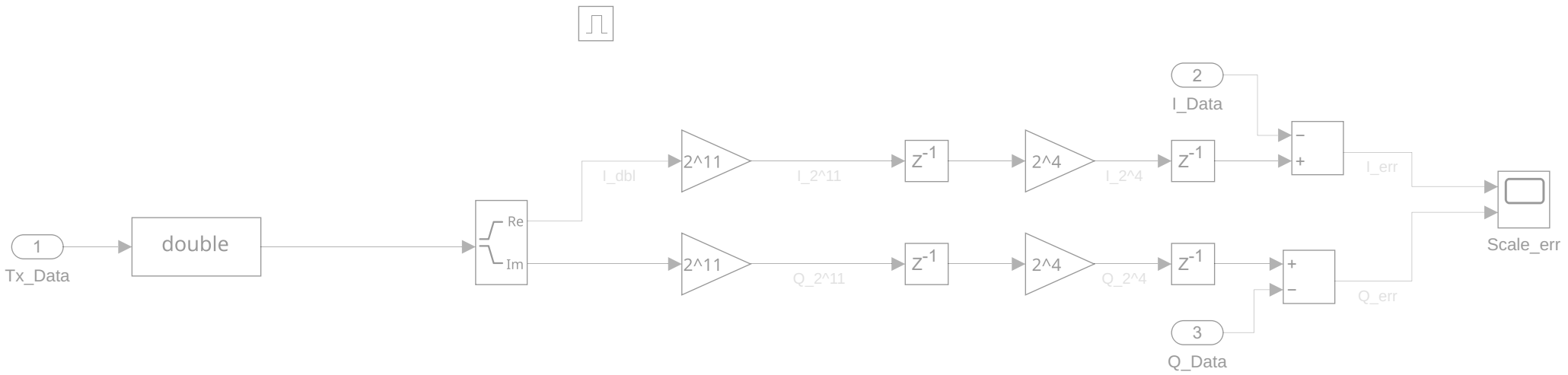
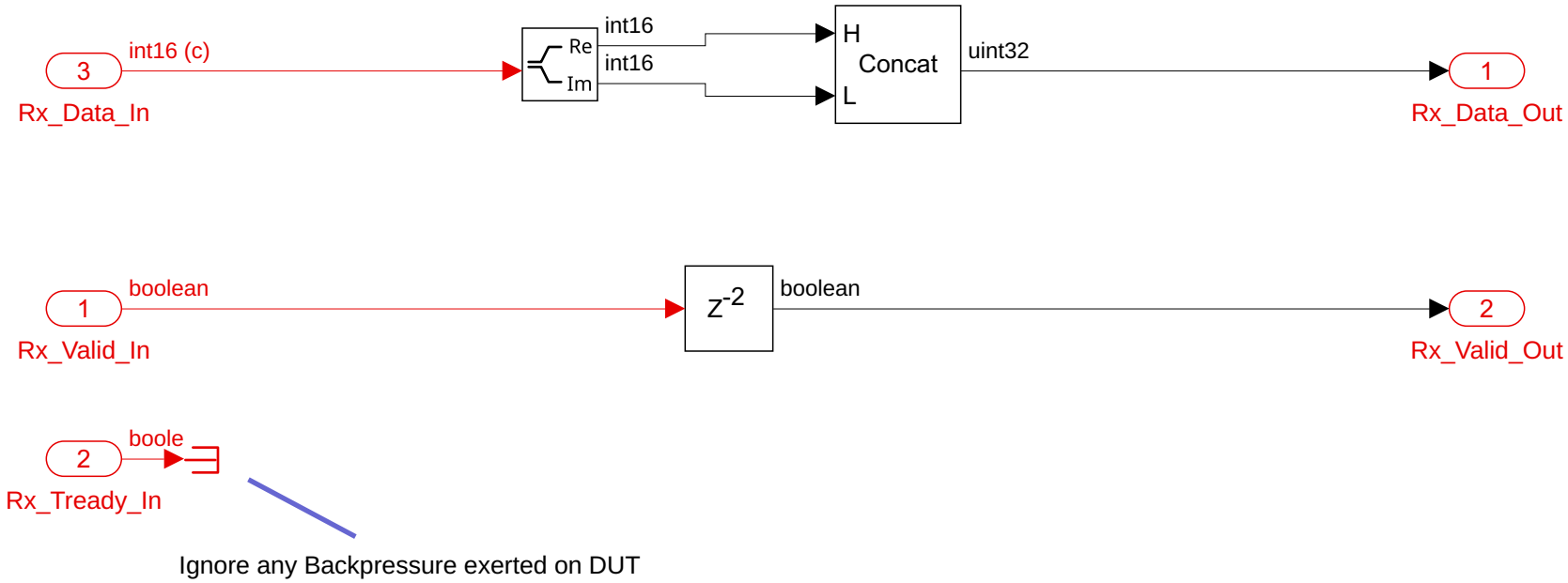
DVB-S2 AXI Stream Transmitter

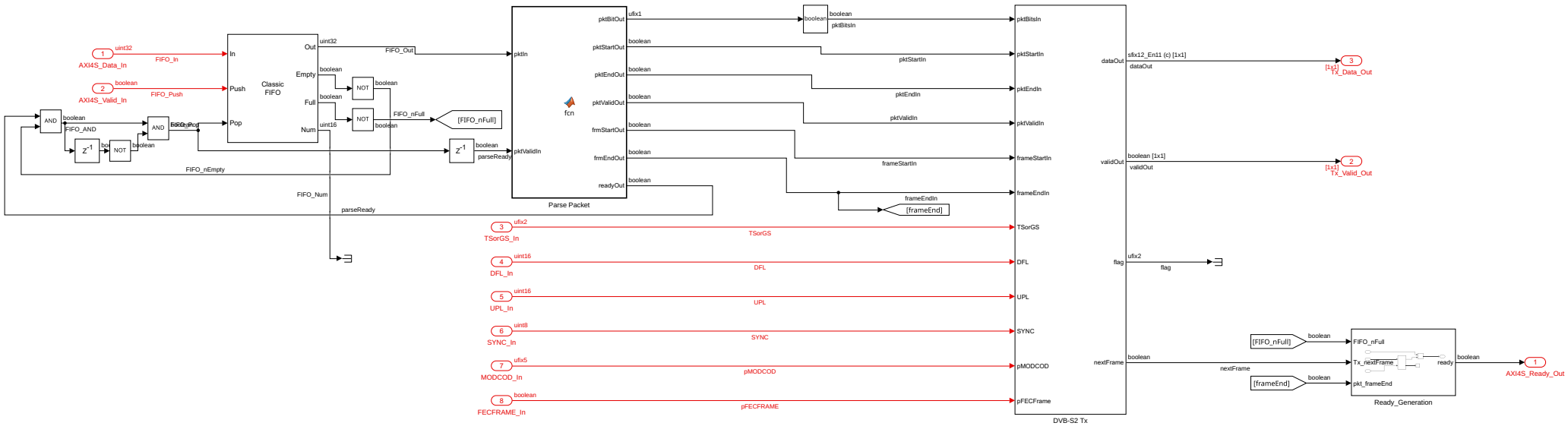



AXI4S_Data_In
AXI4S_Valid_In
DVB-S2 Control Signals
TSorGS_In
DFL_In
UPL_In
SYNC_In
MODCOD_In
FECFRAME_In
Tx_Valid_In
Tx1_Data_I_In
Tx1_Data_Q_In
AXI4S_Ready_Out
Tx_Valid_Out
Tx_Data_Out
Tx
HDL_DUT_Ready
AXI4S_Tready_Out
Shift fraction bits to integer scaled for 12 bit DAC
convert
Bit shift from dialog
Bit shift from dialog1
Tx_Valid_Out
Tx1_Data_I_Out
Tx1_Data_Q_Out
No HDL


Rx_Valid_In
Rx1_Data_I_In
Rx1_Data_Q_In
AXI4S_Tready_In
Rx
Rx1_Data_I_Out
Rx1_Data_Q_Out
Rx2_Data_I_Out
Rx2_Data_Q_Out
Rx_Valid_Out
AXI4S_Data_Out
AXI4S_Valid_Out


1
Tx_Data
double
Re
Im
I_dbl
2^11
I_2^11
Q_2^11
2^11
Z-1
Z-1
Z-1
Z-1
2^4
2^4
I_2^4
Q_2^4
2
I_Data
3
Q_Data
I_err
Q_err
Scale_err

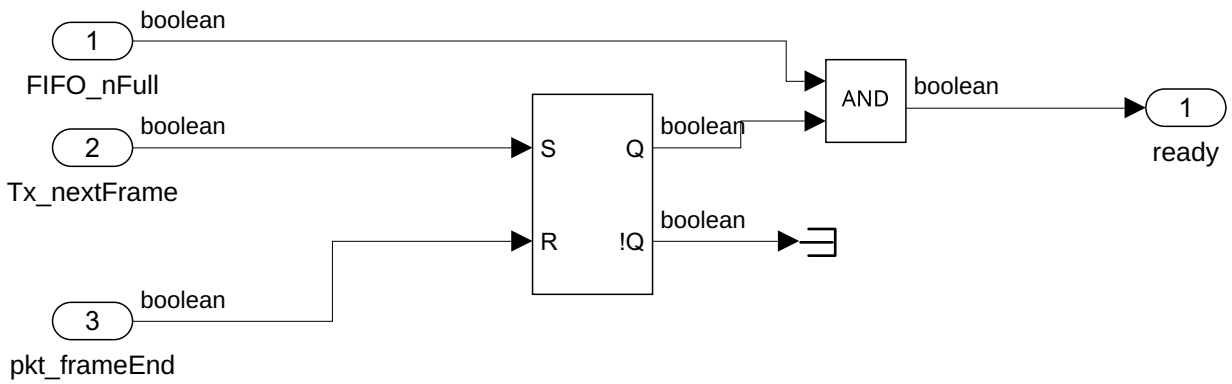

3
int16 (c)
Rx_Data_In
Re
Im
int16
int16
H
Concat
L
uint32
1
Rx_Data_Out
1
boolean
Rx_Valid_In
z-2
boolean
2
Rx_Valid_Out
2
boole
Rx_Tready_In
Ignore any Backpressure exerted on DUT

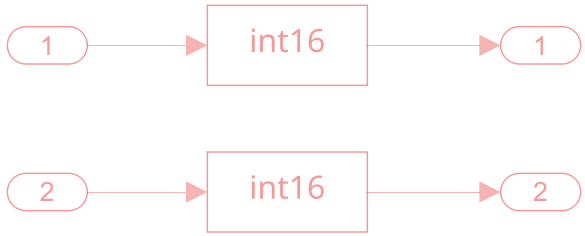

1
uint32
AXI4S_Data_In
FIFO_In
2
boolean
AXI4S_Valid_In
FIFO_Push
In
Push
Pop
Classic FIFO
Out
Empty
Full
Num
uint32
boolean
boolean
uint16
NOT
NOT
boolean
boolean
FIFO_Out
FIFO_nFull
[FIFO_nFull]
AND
boolean
FIFO_AND
AND
NOT
boolean
FIFO_nEmpty
FIFO_Num
parseReady
boolean
parseReady
pktIn
pktValidIn
fcn
Parse Packet
pktBitOut
pktStartOut
pktEndOut
pktValidOut
frmStartOut
frmEndOut
readyOut
ufix1
boolean
pktBitsIn
boolean
pktStartIn
boolean
pktEndIn
boolean
pktValidIn
boolean
frameStartIn
boolean
frameEndIn
boolean
[frameEnd]
3
ufix2
TSorGS_In
TSorGS
4
uint16
DFL_In
DFL
5
uint16
UPL_In
UPL
6
uint8
SYNC_In
SYNC
7
ufix5
MODCOD_In
pMODCOD
8
boolean
FECFRAME_In
pFECFRAME
pktBitsIn
pktStartIn
pktEndIn
pktValidIn
frameStartIn
frameEndIn
TSorGS
DFL
UPL
SYNC
pMODCOD
pFECFrame
DVB-S2 Tx
dataOut
sfix12_En11 (c) [1x1]
dataOut
3
[1x1]
Tx_Data_Out
validOut
boolean [1x1]
validOut
2
[1x1]
Tx_Valid_Out
flag
ufix2
flag
nextFrame
boolean
nextFrame
[FIFO_nFull]
boolean
[frameEnd]
boolean
FIFO_nFull
Tx_nextFrame
pkt_frameEnd
ready
Ready_Generation
boolean
1
AXI4S_Ready_Out

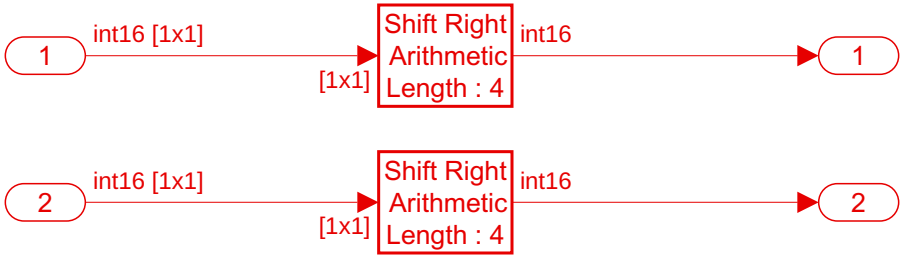
```
% Parse Input TDATA into serialised Byte and control signals
% On Valid In & Ready: Read dataIn
% next 8 cycles: dataIn bits are output on pktBitsOut
% If no valid input: Hold current output state
function [pktBitOut,pktStartOut,pktEndOut,pktValidOut,frmStartOut,frmEndOut,readyOut] = fcn(pktIn,pktValidIn,LSBfirst,frameEnd_pos
    arguments
        pktIn           (1,1) uint32
        pktValidIn      (1,1) logical
        LSBfirst        (1,1) logical
        % Control Signal bit positions
        frameEnd_pos    (1,1) %= 3;
        frameStart_pos  (1,1) %= 2;
        pktEnd_pos      (1,1) %= 1;
        pktStart_pos    (1,1) %= 0;
        % Payload byte start position
        payload_pos     (1,1) %= 4;
    end

    % Persistent Vars
    persistent count;
    persistent packetReg;
    persistent byteInReg;
    % Bit Masks
    frameEnd_mask   = 2^frameEnd_pos;
    frameStart_mask = 2^frameStart_pos;
    pktEnd_mask     = 2^pktEnd_pos;
    pktStart_mask   = 2^pktStart_pos;
    payload_mask    = bitshift( ...
                        (2^8)-1,...
                       payload_pos ...
                       );


    if isempty(count)
        count = fi(1,0,log2(8)+1,0,hdlfimath);
        packetReg = uint32(0);
        byteInReg = fi(0,0,8,0,hdlfimath);
    end

    % Other Vars
    it = fi(1,0,log2(8)+1,0,hdlfimath);

    % Extract bit
    if count == 1
        if pktValidIn
            % Get input packet and load into registers
            packetReg = pktIn;
            % packetReg = 0b[0...][Payload Byte][Flags]
            byteIn = bitshift(...
                            bitand(packetReg,  payload_mask),...
                            payload_pos*-1 ...
                        );
            byteInReg(:) = byteIn;
            % Count from LSB or count from MSB
            if LSBfirst
                it(:) = count;
            else
                it(:) = 8 - (count-1);
            end
```

```
        % Parse packet and frame start values
        pktStartOut = logical(bitshift(...
                        bitand(packetReg,  pktStart_mask),...
                        pktStart_pos*-1 ...
                        ));
        frmStartOut   = logical(bitshift(...
                        bitand(packetReg,  frameStart_mask),...
                        frameStart_pos*-1 ...
                        ));
        % End signals never true during bit serialisation
        [pktEndOut,frmEndOut] = deal(false);
        readyOut = true; % Keep ready for 1 cycle
        pktValidOut = true;
        pktBitOut = bitget(byteInReg,it);
        count(:) = count + 1;
    else
        % No Valid Data
        % Set Ready, wait for valid data
        count(:) = 1;
        it(:) = 1;
        readyOut = true;
        pktValidOut = false;
        pktBitOut = bitget(byteInReg,it);
        [pktStartOut,pktEndOut,frmStartOut,frmEndOut] = deal(false);
    end
elseif count < 8 && count > 1
    % Serialising bits
    if LSBfirst
            it(:) = count;
        else
            it(:) = 8 - (count-1);
    end
    pktBitOut   = bitget(byteInReg,it);
    pktValidOut = true;
    % Flag signals never true during bit serialisation
    [pktStartOut,pktEndOut,frmStartOut,frmEndOut] = deal(false);
    readyOut    = false;
    count(:)    = count + 1;
elseif count == 8
    if LSBfirst
            it(:) = count;
        else
            it(:) = 8 - (count-1);
    end
    % Parse packet and frame end values
    pktEndOut = logical(bitshift(...
                    bitand(packetReg,  pktEnd_mask),...
                    pktStart_pos*-1 ...
                    ));
    frmEndOut   = logical(bitshift(...
                    bitand(packetReg,  frameEnd_mask),...
                    frameEnd_pos*-1 ...
                    ));
    % Start values never true at end of payload byte
    [pktStartOut,frmStartOut] = deal(false);
    pktBitOut   = bitget(byteInReg,it);
    pktValidOut = true;
    readyOut    = true;
    count(:)    = 1;
```

```
    else
        count(:) = 1;
        it(:) = 1;
        readyOut = false;
        pktValidOut = false;
        pktBitOut = bitget(byteInReg,it);
        [pktStartOut,pktEndOut,frmStartOut,frmEndOut] = deal(false);
    end
end
```


1
boolean
FIFO_nFull
2
boolean
Tx_nextFrame
3
boolean
pkt_frameEnd
S
R
Q
!Q
boolean
boolean
AND
boolean
1
ready


1
int16
1
2
int16
2


1
int16 [1x1]
[1x1]
Shift Right
Arithmetic
Length : 4
int16
1
2
int16 [1x1]
[1x1]
Shift Right
Arithmetic
Length : 4
int16
2

Sample Times for 'dvbs2_stream'

Color	Annotation	Description	Value
	D1	Discrete 1	1.0000e-06
	D2	Discrete 2	2.0000e-06
	D3	Discrete 3	8.0000e-06
	D4	Discrete 4	1.6000e-05
	D5	Discrete 5	2.5000e-03
	Inf	Constant	Inf
	M	Multirate	N/A