VLSI IP
OWNED BY AVIRAL MITTAL

Mp3HufDec

DATA SHEET AND USER GUIDE

Mp3HufDec

ISO 11172-3 LAYER III HUFFMAN DECODER.

ISO 11172-3 Layer III or commonly called as Mp3, employs Huffman encoding technique to compress data, along with other compression schemes. Our product Mp3HufDec is designed to 'decode' the Huffman coded data present in an Mp3 encoded file. Its written using VHDL, and is fully synthesizeable.

Main Features:

- Synthesizeable single clock VHDL RTL core, with testbench.
- No memory Decoder : it dos NOT use any memory to decode the incoming bit stream
- No latency Decoder: O/p values are produced at the very next clock edge, as the corresponding input is received by the decoder.
- Low Gate Count : Less than 5K gates on TSMC 0.18u Tech
- Designed using 'HufGen' : our tool for generic Huffman Decoder Design
- Low licensing Cost: $4000 for Single Use Source.

What is the Huffman Decoder supposed to do?

As per the specifications, of ISO 11172-3 Layer III, the frequency spectrum of the audio signal is packed in the bit stream as 2 granules per frame for each channel(the number of channels can be either 1 or 2), and each granule is consist of 576 frequency samples, or frequency lines. Out of these 576 frequency lines, 'big_values' pairs of lines are coded using one of the 17 unique(32 in all) Huffman tables, 'count1' quadruples are coded using one of the 2 huffman tables for the quadruples, and the rest(576 – 2 x big_values – 4 x count1) are zeros, they are also called 'rzeros'. The task of the Huffman decoder is to decode these frequency lines, and produce corresponding values with their signs. The coding of the frequency lines represented by 'big_values' has a special field associated with them called the 'linbits. 'linbits' number of bits will may follow a 'big_value' if, the magnitude of the corresponding 'big_value' is equal to 15. Value of 'linbits' are given in the specs with each Huffman table. The bit stream syntax containing the Huffman codes is given in the ISO specifications, under the function 'Huffmancodebits()'. The task of the Huffman decoder is to produce these 576-rzeros frequency lines for each granule with their correct sign and magnitude. For each Huffman coded 'big_value', there are 2 decoded frequency lines called 'is_x_out' and 'is_y_out'. For each Huffman coded 'count1', there are 4 decoded frequency lines called 'is_v_out_quad', is_w_out_quad', 'is_x_out_quad', 'is_y_out_quad'. The presence of a valid 'is_x_out' and 'is_y_out' is indicated by huf_output_valid. The presence of 'is_v_out_quad' , 'is_y_out_quad' , 'is_x_out_quad' , 'is_y_out_quad' is indicated by 'huf_output_valid_quad' .

The Interface to the Huffman decoder :

Pin Name	IN/OUT	Description.
clk_in	IN	Clock input for the design
clr_n	IN	Async reset for the design when '0', put the design in reset, when '1', let the design work.
mp3bit_stream	IN	Huffman coded bitstream as per Huffmancodedbits()
start	IN	Indicates valid Huffman data corresponding to '*big_values*' part of the spectra, is valid at 'mp3bit_stream'. This signal must be put to '1' to indicate that the current bit stream on the input corresponds to the 'big_values' region of the spectra.
start_quad	IN	Indicates valid Huffman data corresponding to 'count1' part of the spectra is valid at 'mp3bit_stream'. This signal must be put to '1', to indicate that the current bit stream on the input corresponds to the 'count1' region of the spectra
table_sel	IN(4:0)	A 5 bit input signal which indicates which of the 32 tables is to be used for decoding 'big_values'.
count1_tablesel	IN	Signal bit input signal, to indicate which one of the 2 quadruple tables is to be used for decoding 'count1' values
found	OUT	It goes to '1', and indicates that the Huffman decoder has just found a code word in the input bit stream(mp3bit_stream signal). This code word belongs to the 'big_values' region of the spectra
found_quad	OUT	It goes to '1', and indicates that the Huffman decoder has just found a code word in the input bit stream(mp3bit_stream signal). This code word belongs to the 'count1' region of the spectra
is_x_out	OUT(13:0)	The decoded 'x' value corresponding to a 'big_value' Huffman code
is_y_out	OUT(13:0)	The decoded 'y' value corresponding to a 'big_value' Huffman code

sign_x	OUT	Sign bit for 'is_x_out'
sign_y	OUT	Sign bit for 'is_y_out'
is_v_out_quad	OUT(3:0)	The decoded 'v' value corresponding to a 'count1' Huffman code.
is_w_out_quad	OUT(3:0)	The decoded 'w' value corresponding to a 'count1' Huffman code.
is_x_out_quad	OUT(3:0)	The decoded 'x' value corresponding to a 'count1' Huffman code.
is_y_out_quad	OUT(3:0)	The decoded 'y' value corresponding to a 'count1' Huffman code.
sign_v_quad	OUT	Sign bit for is_v_out_quad
sign_w_quad	OUT	Sign bit for is_w_out_quad
sign_x_quad	OUT	Sign bit for is_x_out_quad
sign_y_quad	OUT	Sign bit for is_y_out_quad
huf_output_valid	OUT	Indicates the presence of two valid decoded values 'is_x_out' and 'is_y_out' belonging to the 'big_value' region of spectra. When '1', then 'is_x_out' and 'is_y_out' contains valid decoded values
huf_output_valid_quad	OUT	Indicates the presence of 4 valid decoded value belongs 'is_v_out_quad', is_w_out_quad', 'is_x_out_quad' and 'is_y_out_quad' belonging to the 'big_value' region of spectra. When '1', then 'is_v_out_quad', is_w_out_quad' 'is_x_out_quad' and 'is_y_out_quad' contains valid decoded values

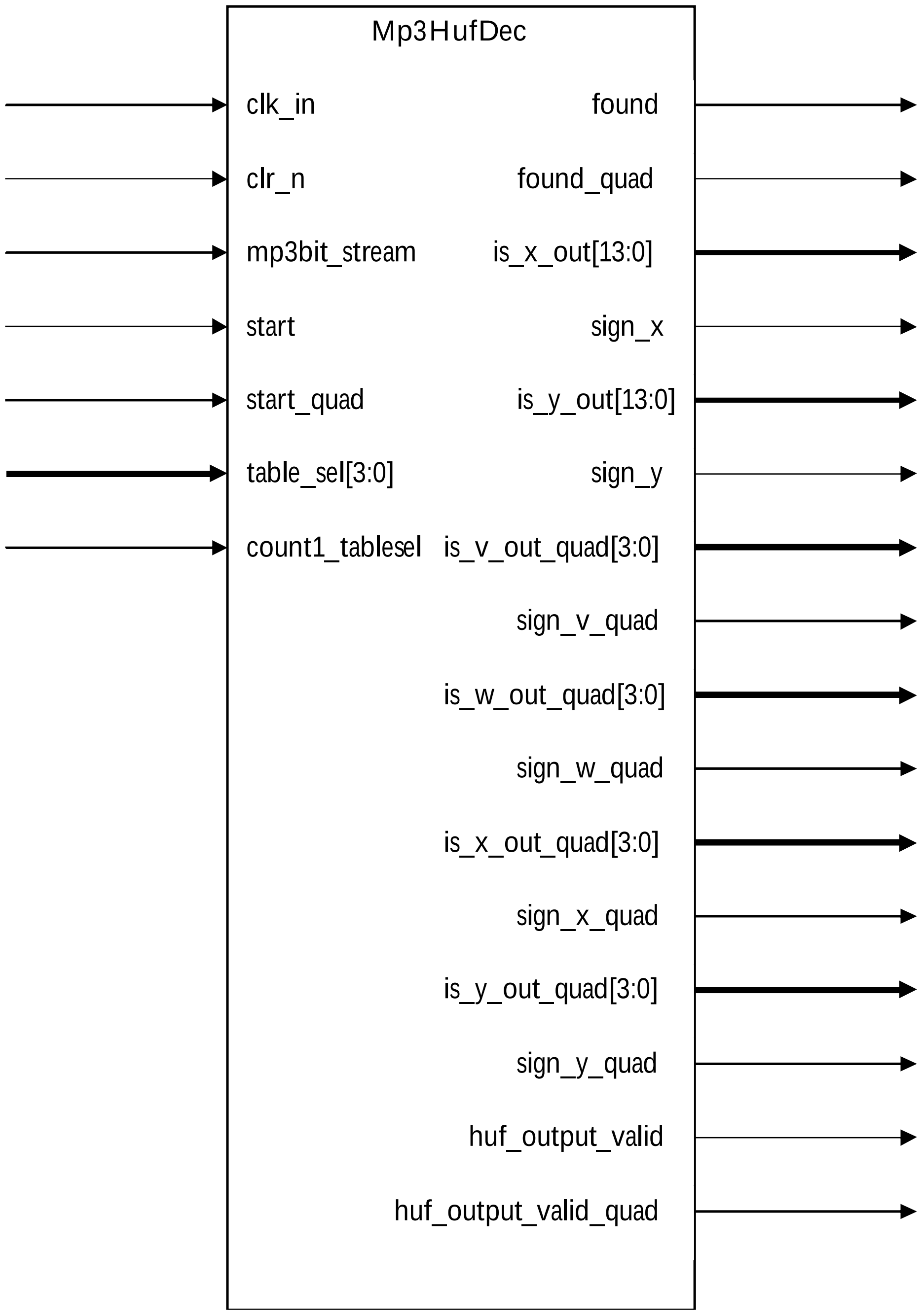

Figure 1: The Entity for Huffman Decoder with its I/Os.

Section 2.

Working of the Huffman Decoder.

It is strongly recommended that the reader should go through the syntax of huffmancodedbits(), in appendix A, before reading this section.

As soon as one of the inputs start or start_quad goes to '1', and clr_n, is not enabled i.e its set to '1', and there is a running clock at the clk_in input, Huffman decoder starts working. As soon as the input bitstream matches, a Huffman code from the table given by 'table_sel' (in case start is '1' and start_quad is 0), 'found' goes '1' in the same clock cycle. As per the syntax of huffmancodedbits(), depending upon the corresponding decoded values, i.e x,y from the table, the sign and the linbits corresponding to each decoded value(x and y) follows hcod(x,y) in a manner described in huffmancodedbits(). The elements that may follow a hcod are

linbits(x), if |x| = 15 and linbits !=0, for the table corresponding to table_sel

Sign(x), if |x| != 0

linbits(y), if |y| = 15 and linbits !=0, for the table corresponding to table_sel

sign(y), if|y| != 0

After all the expected elements corresponding to a hcod, are received by the Huffman decoder, it produces final values 'si_x_out' and 'si_y_out' along with their sign 'sign_x' and 'sign_y', and the output signal 'huf_output_valid' is put to '1'. The user can use 'huf_output_valid' signal to clock in the data from the Huffman decoder.

Figure 2 below gives an example of Huffman decoding using Mp3HufDec, with 2 codes being decoded from table number 24(linbits = 4). The first code word(hcod|x|,|y|) being decoded is '0010000' which corresponds to x=3,y=15. Since |x| is not equal to 15, no linbits for 'x' are expected. Since 'x' is not equal to 0, therefore a sign bit for 'x' is expected immediately after hcod(|x|,|y|). Since |y| is equal to 15, therefore linbits for 'y'(linbitsy, 4 bits) are expected immediately after sign(x). Since |y| not equal to 0, sign bit for 'y' is expected immediately after 'linbitsy'(shown to be equal to 0011 in Figure 2). The final value of x is therefore -3, and final value for y therefore +18, (which comes from |y| + linbitsy).(Remember sign bit = 1 => -ive number, and sign bit = 0 => +ive number) Similarly it can be seen that the second hcod(|x|,|y|), which is shown to be '1100' is followed by just sign(x), sign(y), giving x = +1, and y = +1. The decoding of the frequency spectrum represented by 'count1' is also decoded in similar fashion as the 'big_values' region, but in 'count1' region, there are no linbits and instead of juxt 'x' or 'y', there are four decoded values instead of 2 i.e 'v', 'w', 'x', 'y'. The final values are called 'is_v_out_quad', 'is_w_out_quad' , 'is_x_out_quad', 'is_y_out_quad' with their sign bits as 'sign_v_quad', 'sign_w_quad', 'sign_x_quad' , 'sign_y_quad'. These values are qualified by a presence of 'huf_output_valid_quad'. Note that only one of the outputs, either 'huf_output_valid' or

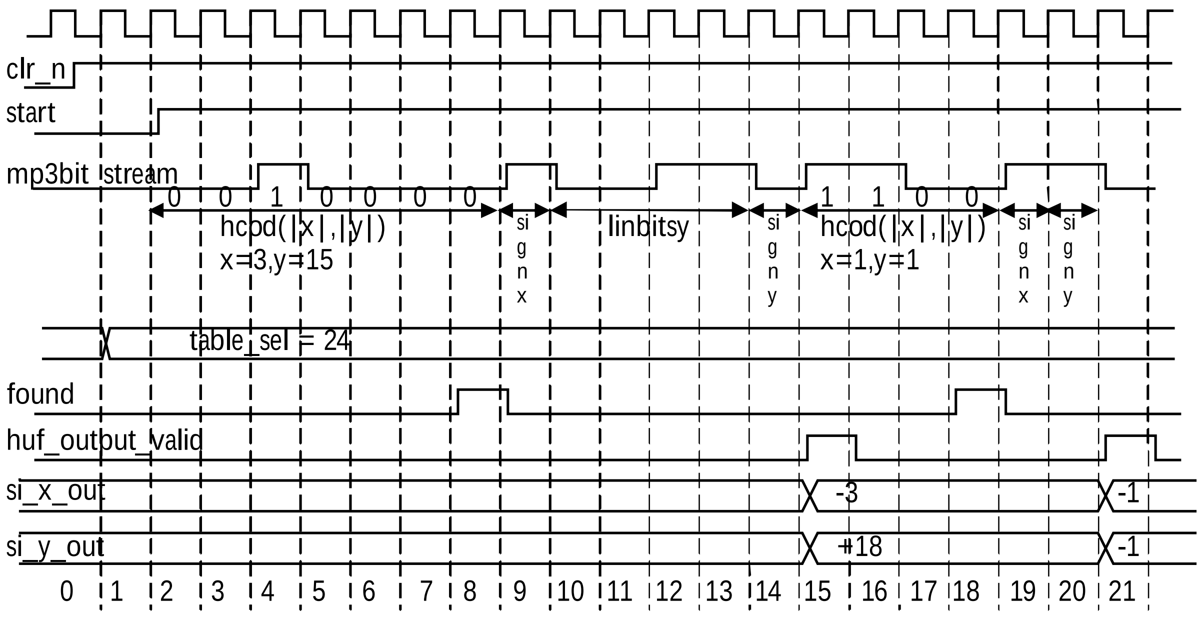

Figure 2. An example, where the huffman decoder is decoding a bit stream, having codes from Table Number 24(linbits = 4).

'huf_output_valid_quad' can be active at any given time, provided, the inputs 'start' and 'start_quad' are not applied simultaneously.

The outputs 'found' and 'found_quad' are very helpful for user, as these can be used by user to keep an account of a 'counter' which counts how many decoded values have been received by user. For example if there exists a counter called 'huf_codes_coutner' then this counter should be incremented by 2, whenever 'found' goes to '1', and it should also be incremented by 4, whenever 'found_quad' goes high for each clock cycle. If either 'found' or 'found_quad' is '1' continuously for 'n' clock cycles, where n>1, then it definitely means that the 'huf_codes_counter' should be incremented by

Case I: 2 for each number of clock cycles, 'found' was '1', i.e after the end of 'n' clock cycles, the 'huf_codes_counter' should have been incremented by n*2.

Case II: 4 for each number of clock cycles 'found_quad' was '1', i.e after the end of 'n' clock cycles, 'huf_codes_counter' should have been incremented by n*4.

For example if 'table_sel' is 16, 'start' is '1', and 'mp3bit_stream' is '1' for 'n' clock cycles. It means that the input bit stream is consist of 'n' code words, corresponding to first entry in the table where hcod(|x|,|y|) = '1'.

User should update the value of 'table_sel' if region boundary has been discovered, i.e 'huf_code_counter' = (number of codes in a region -2) and 'found' has been received. For example, in the 'big_values' region of the spectrum if number of codes in region0 are 48, then if huf_code_counter reaches 46, and a 'found' has been asserted, user should switch the value of 'table_sel' to point the correct 'table_sel' for region1. The new 'table_sel' will not come into affect, until all the sign bits and linbits of 24th hcod has been received by the Huffman decoder. This example is also illustrated in Figure 3 below. Note that even after 'table_sel' has been updated in clock period number 9, soon after 'found' has been detected, the final values i.e 'si_x_out' and 'si_y_out' produced as output in clock period number 15, belongs to table Number 24(identified by previous 'table_sel' value). Hence from this point of time the 'si_x_out' and 'si_y_out' will belong to table Number 15(identified by new 'table_sel' value. The values 'si_x_out' = 0, and 'si_y_out' = -1 produced in clock period number 21, thus belongs to table number 15.

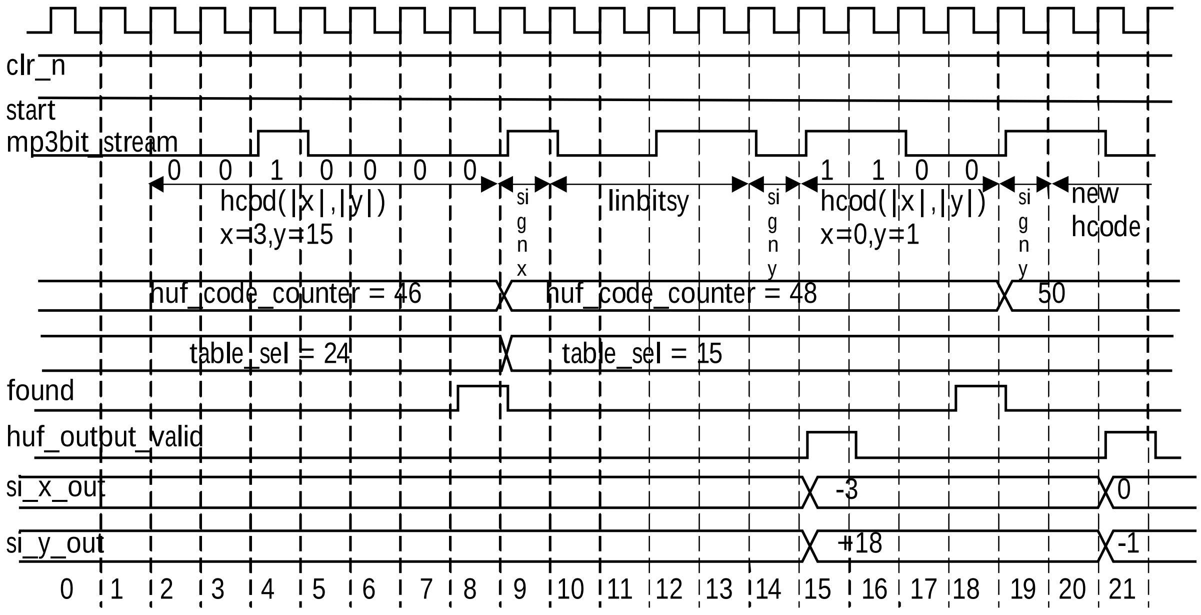

Figure 3. **An example, to show when the 'table_sig' must be updated, if the number** of codes reaches a region boundary.

Section 3.

Testing of the Huffman decoder.

Testing of the Huffman decoder is done using various test cases.

- Test Case I. This test case exercises each hcod present in each of the 17 unique Huffman tables for 'big_values' region, and each hcod present in both the quad tables, given in the specifications. This test case is called EXHAUST1. Corresponding testbench is 'huffman_tb_exhaust1.vhdl'

- Test Case II. This test case exercises hcod form some real .mp3 files.