The Huffman decoder Core

publish at www.opencores.org

written by René Doss

(Dossmatik GmbH/ Germany)

Version 0.1/ Date 20.12.2009

Disclaimer:

Dossmatik is not responsible for the end product or the success of the end product in any way. The final sign and acceptance of any products or designs is the sole responsibility of the user. This core is at the moment not full tested. The actual state is alpha phase.

Introduction:

In the follow document a decoder is described for baseline jpeg pictures. The code is written for real time video streaming. Some efforts are made in optimization for speed and dynamic huffman table and dynamic quantization tables load.

Language: VHDL

License:

If you use this code for simulation there are only one restrictions. You have to mention the core origin in your documentations and publications in a valuable position. Of cause you can send me also a publication.

If you fit this code into hardware components like FPGA, ASIC or analogous proceeding then is the license **_GPL_**.

It is not allowed to put this code with commercial code together in hardware. If you have a problem with this restriction you can contact me.

Generally explanation:

This core analyses jpeg baseline makers is the incoming data stream. A state machine switch in the correct state. The incoming picture header information is analyzed and is applied in the decoding process. All tables are dynamic.

Actual are this:

0xFFD8	Start of image
0xFFE0	App0 application segment
0xFFDB	define quantization table
0xFFC0	SOF0 Baseline DCT
0xFFC4	define huffman table
0xFFDA	start of scan
0xFFD9	End of image

Interface:

entity huffman_decoder is

port(

clk   :in std_logic;

--interface data input

wr   :in std_logic;   --write

data_in   : in unsigned (7 downto 0);   --data jpeg stream input

wr_en   : out std_logic:='1';   --write enable

--interface data out

output_valid   : buffer std_logic;   --use it as write signal in the follow IDCT

data_out   : out signed (15 downto 0);   --decoded and dequantized coefficient

next_eob   : buffer std_logic:='0';   --the next data is the last coefficient of block

                                                        --all higher zigzag coeficients are zero

sop   : out std_logic:='0';   --start of picture

eop   : out std_logic:='0';   --end of picture
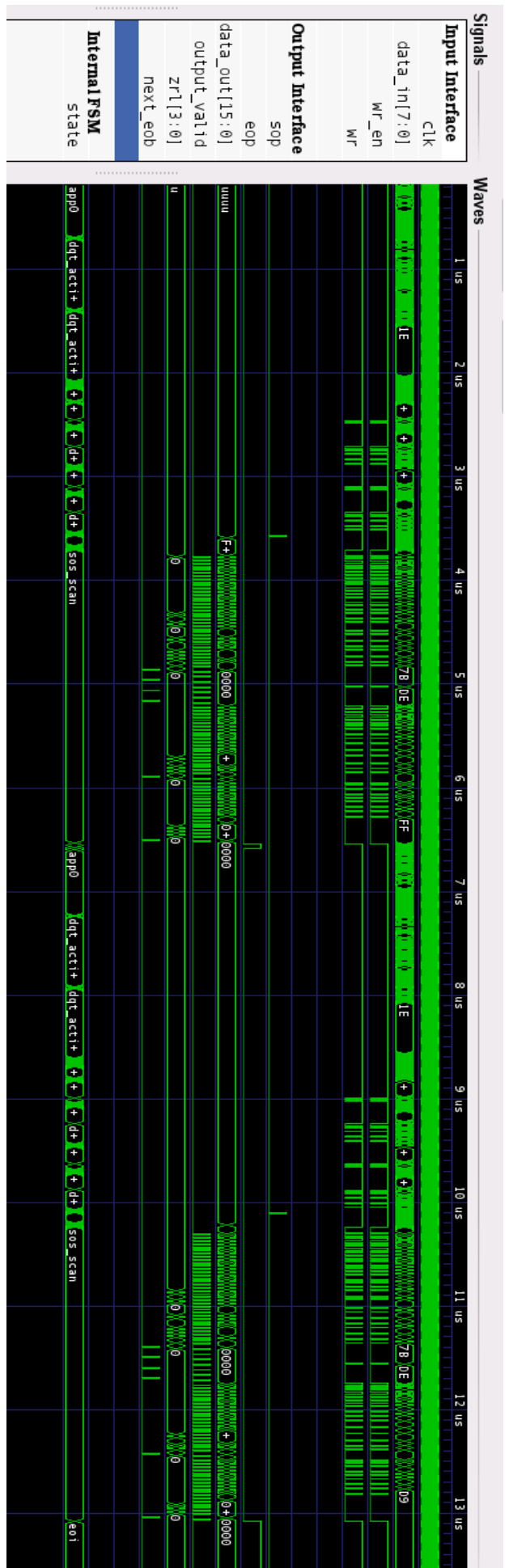
zrl   : out unsigned (3 downto 0);   -- number of consecutive zeros before the next
                                                        --coefficient

decoder_enable   : in std_logic);

end huffman_decoder;

first picture:

header information
- quantization tables

- huffman tables

Pixel information in the stream:
sop (start of picture)   goes high

  switch to the valid table
  decode and dequantizered
  output the value in zigzag order

eop (end of picture) goes high

second picture:

header information
- quantization tables

- huffman tables

Pixel information in the stream:
sop (start of picture)   goes high

  switch to the valid table
  decode and dequantizered
  output the value in zigzag order

eop (end of picture) goes high