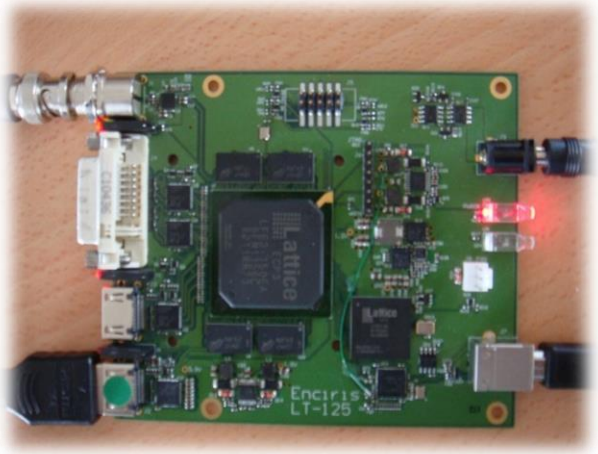
Video decoding: SDI interface implementation & H.264/AVC bitstream decoder hardware architecture design and implementation



Student Name:	Mr. Vicheka PHOR
Department:	CAMSI
School supervisor:	Mr. Jacques JORDA
Advisor:	Mr. Phillip WEISSFLOCH
Academic year:	2013-2014

OUTLINE

I. Objectives

II. SDI video interface implementation

- II.1 Tri-Rate SDI PHY IP Pass-through sample design
- II.2 SDI video interface implementation on LT-125 board

III. H.264/AVC bitstream decoder hardware architecture design and implementation

- III.1 H.264/AVC data stream structure
- III.2 H.264/AVC bitstream decoder hardware architecture
- III.3 FLC, Exp-Golomb, CAVLD decoding
- III.4 Development progress
- III.5 Tools used in the FPGA development flow design
- III.6 Simulation and results

IV. Conclusion

I. Objectives

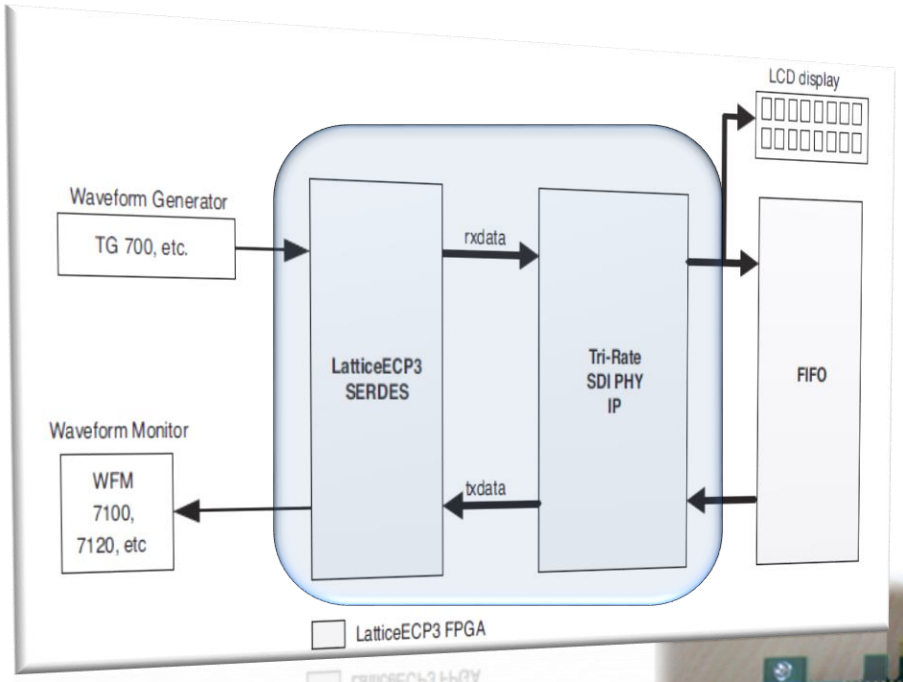
- SDI (Serial Digital Interface) video interface implementation:
  - implementation of a Lattice tri-rate SDI PHY IP core on a Lattice FPGA of the Enciris LT-125 board

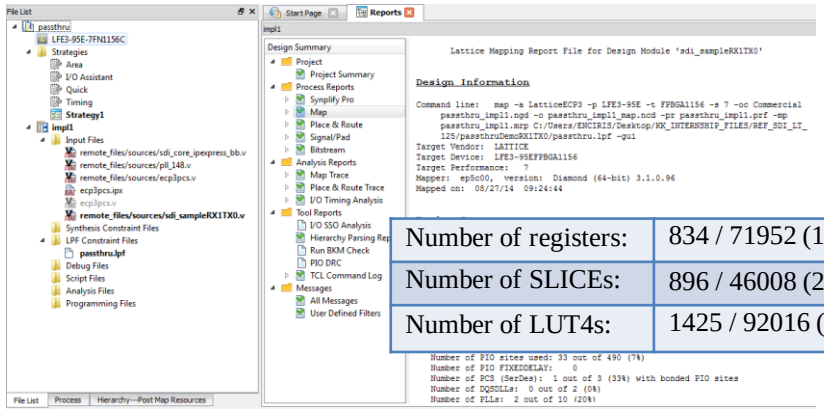
- H.264/AVC bitstream decoder hardware architecture design and implementation

II. SDI video interface implementation

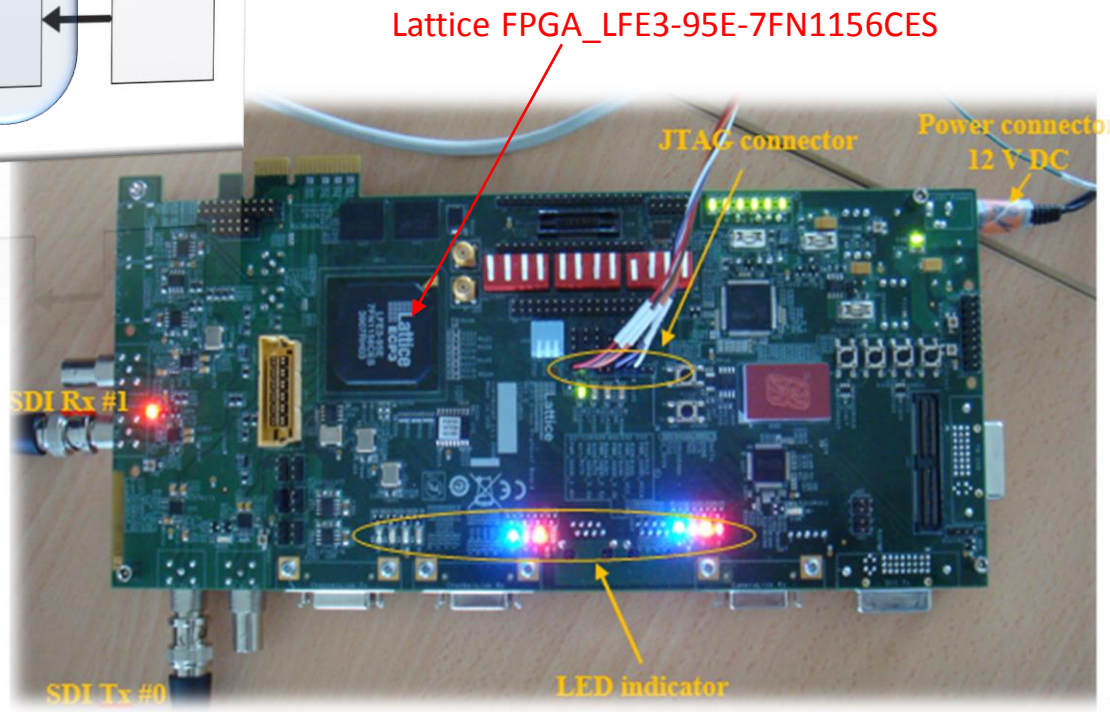
- Objective:
  - 3G/HD/SD SDI video input functionality on LT-125 board
- What was done:
  - Use of Lattice tri-rate SDI PHY IP core
  - Study of the Lattice tri-rate SDI PHY IP Pass-through demo
  - Adding of several blocks: Sync-signals, sdi_422to444, and YCrCb to RGB converter block

II.1 Tri-Rate SDI PHY IP Pass-through sample design

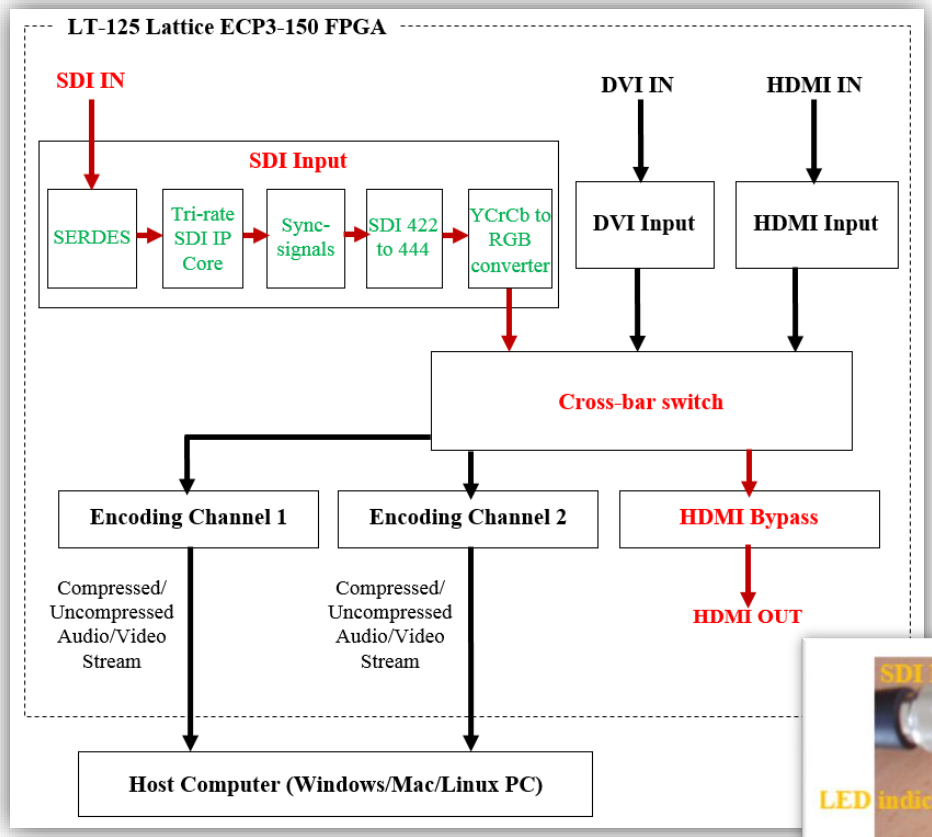


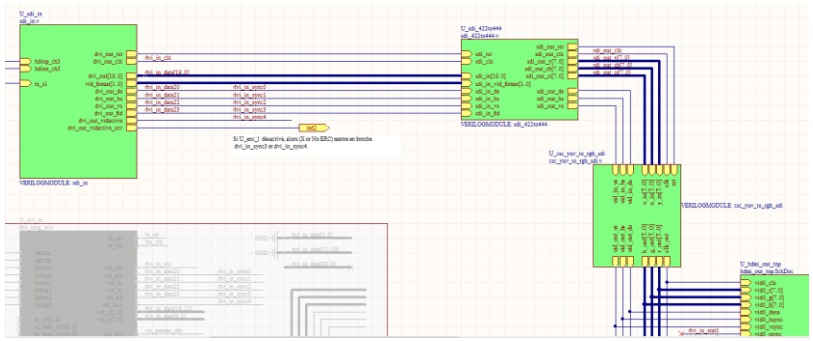


Number of registers:	834 / 71952 (1 %)
Number of SLICEs:	896 / 46008 (2 %)
Number of LUT4s:	1425 / 92016 (2 %)

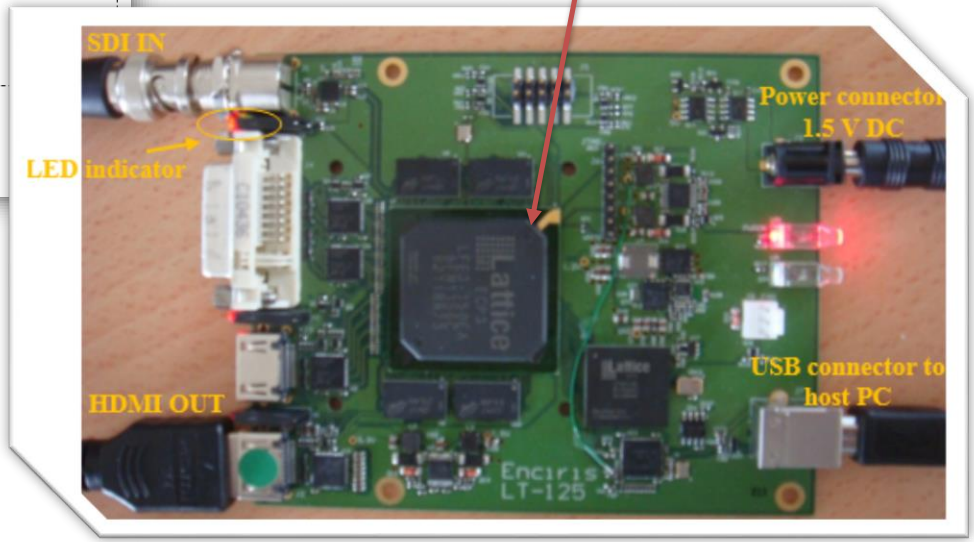


II.2 SDI video interface implementation



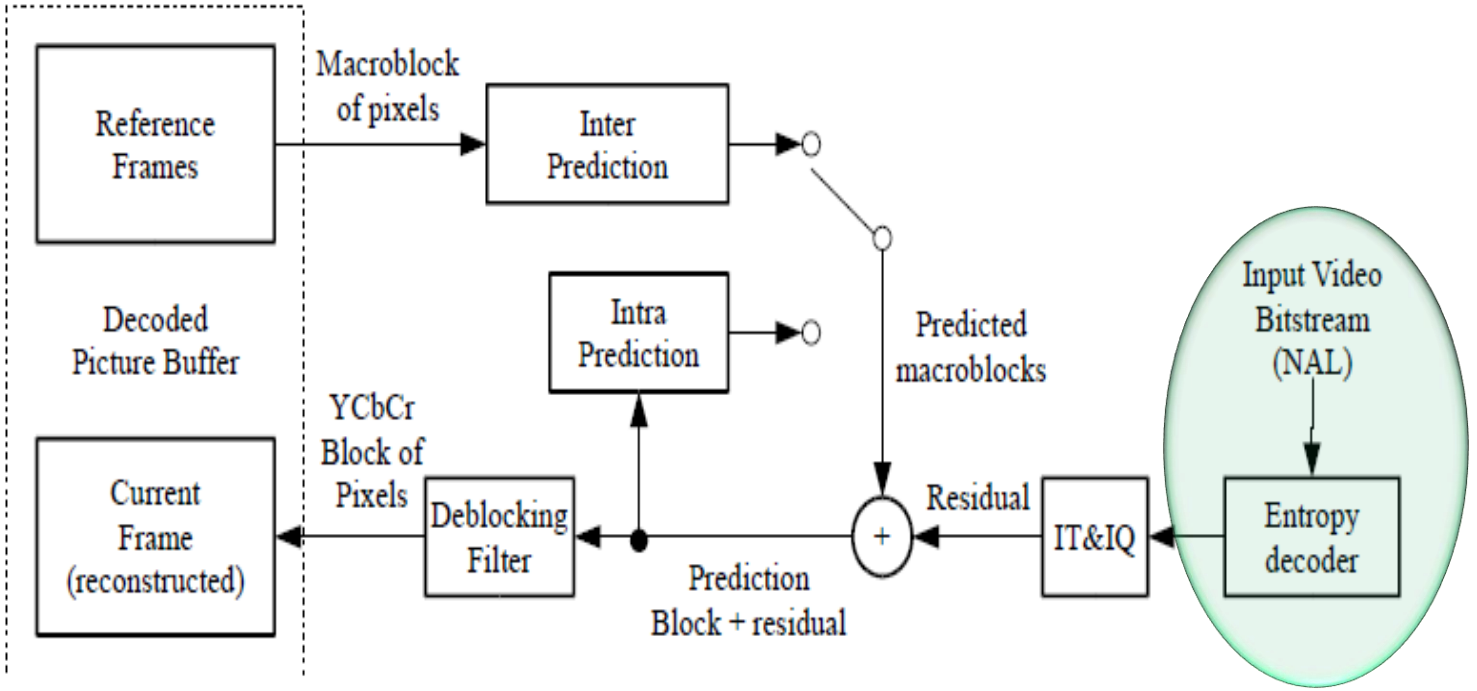


Lattice FPGA_LFE3-150EA-6FN1156C

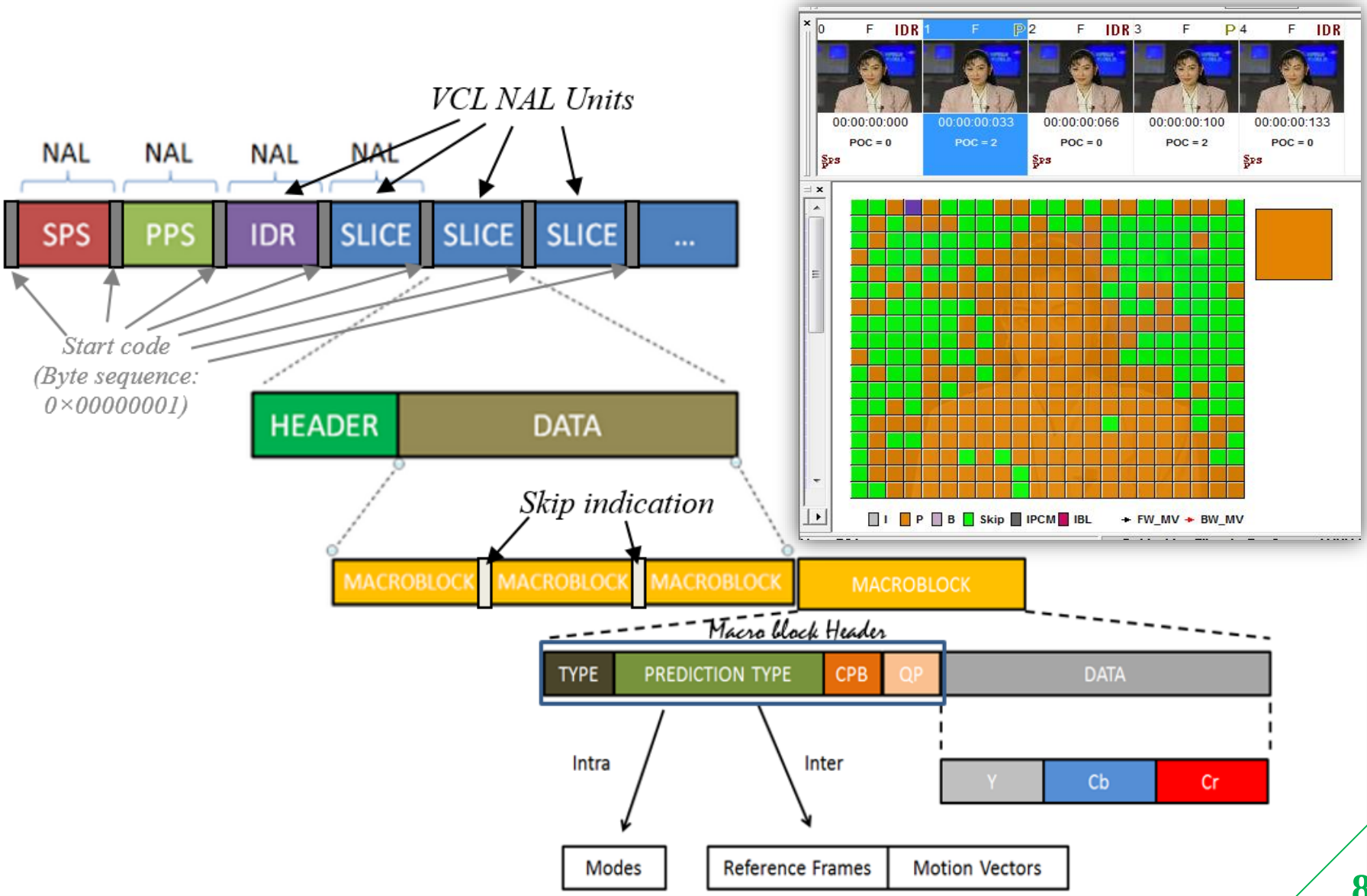


III. H.264/AVC bitstream decoder hardware architecture design and implementation

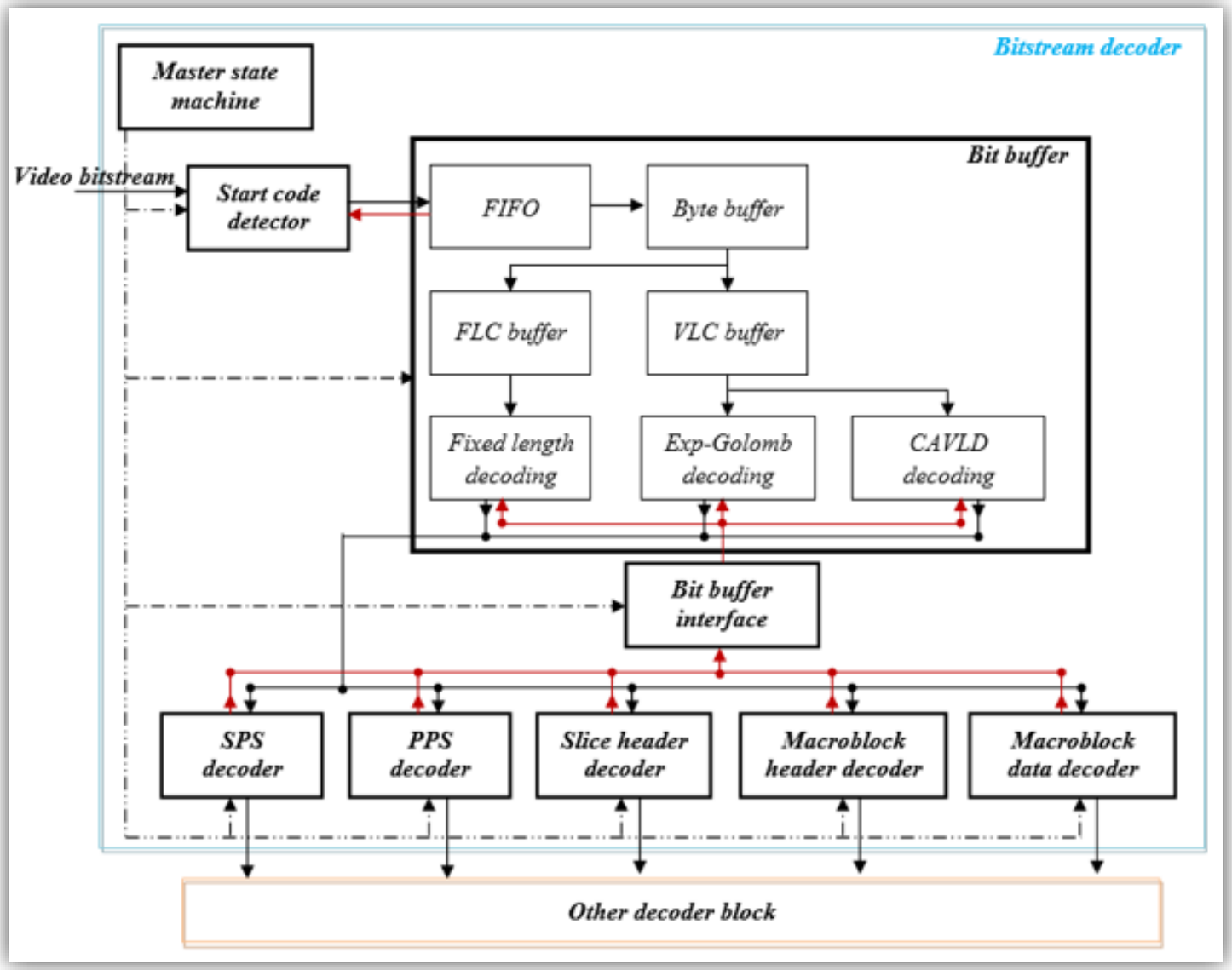
- Objective:
  - develop a custom H.264 bitstream decoder for Enciris encoder (high bit rate: 50 Mbit/s; 1080p30)



III.1 H.264/AVC data stream structure



III.2 H.264/AVC bitstream decoder hardware architecture



III.3 FLC, Exp-Golomb, CAVLD decoding

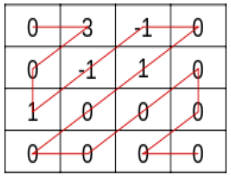
FLC (5 bits)	
code_num	Codeword
0	00000
1	00001
2	00010
3	00011
4	00100
5	00101
6	00110
7	00111
8	01000
...	...

Exp-Golomb	
code_num	Codeword
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001
...	...

CAVLC/CAVCD :
different sets of variable-length codes are chosen depending on the statistics of recently-coded coefficients

16 coefficients → 24 bits *CAVLC*

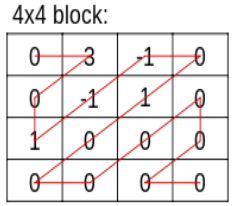
4x4 block:



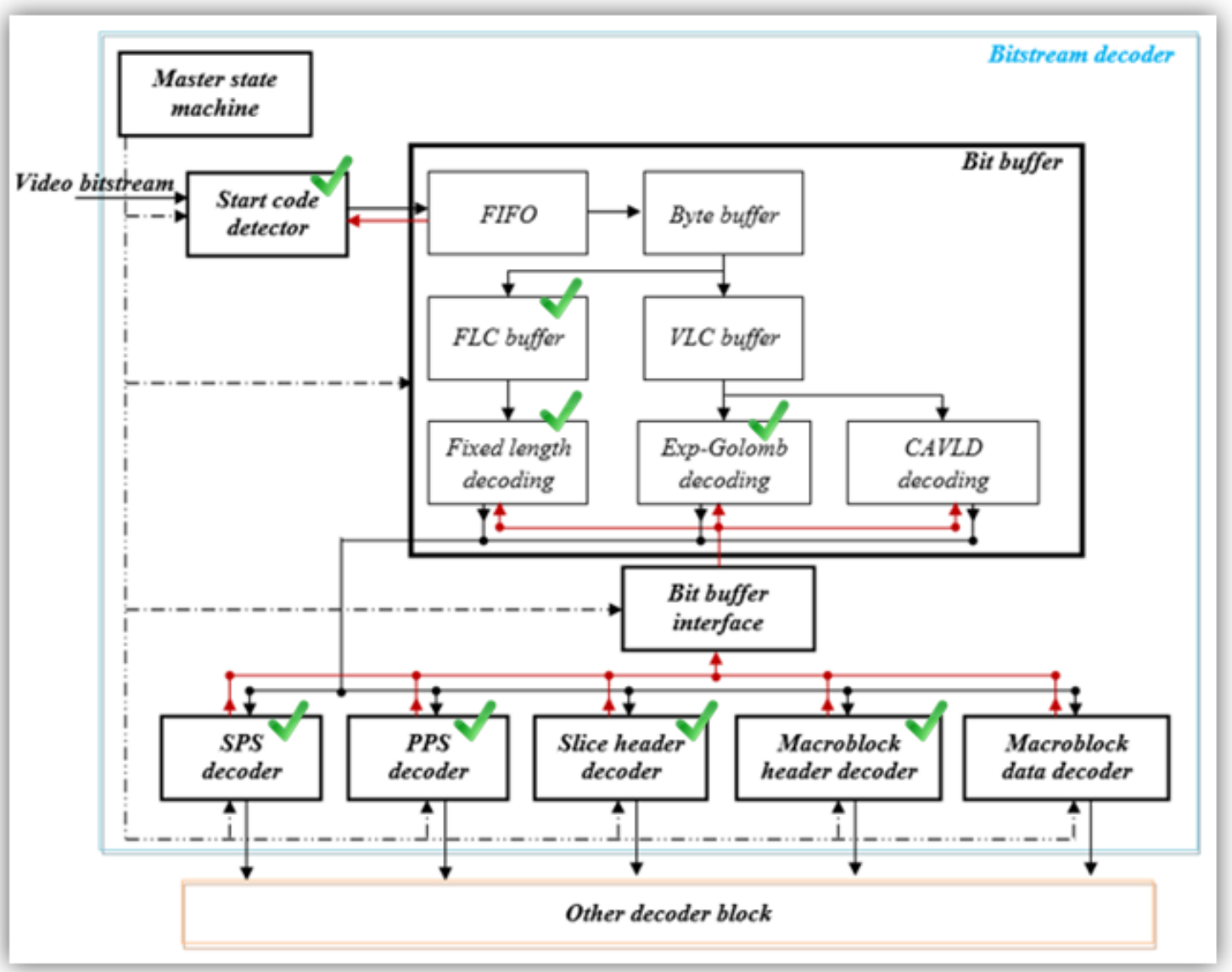
→ 000010001110010111101101

24 bits →16 coefficients *CAVLD*

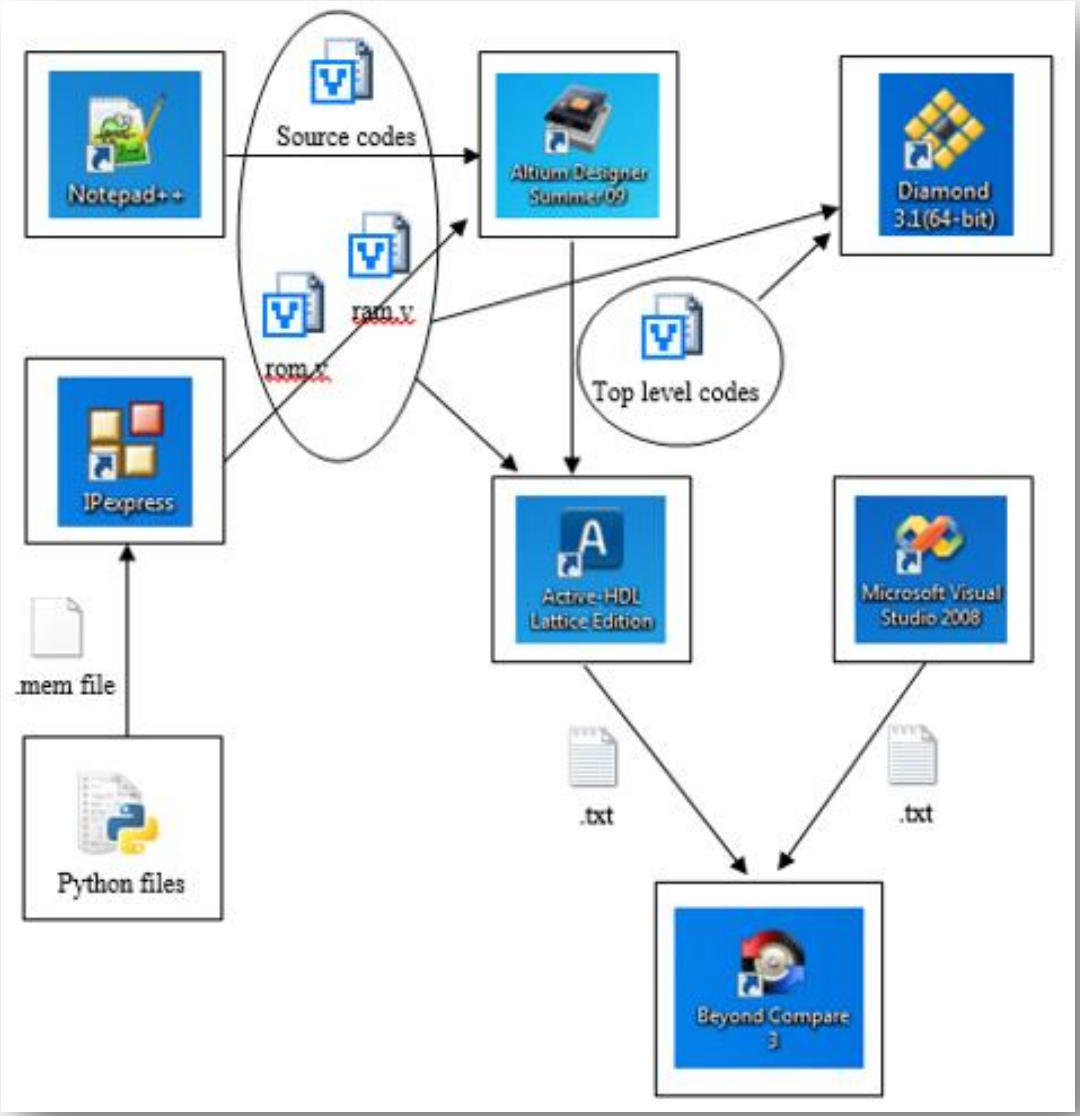
000010001110010111101101 →



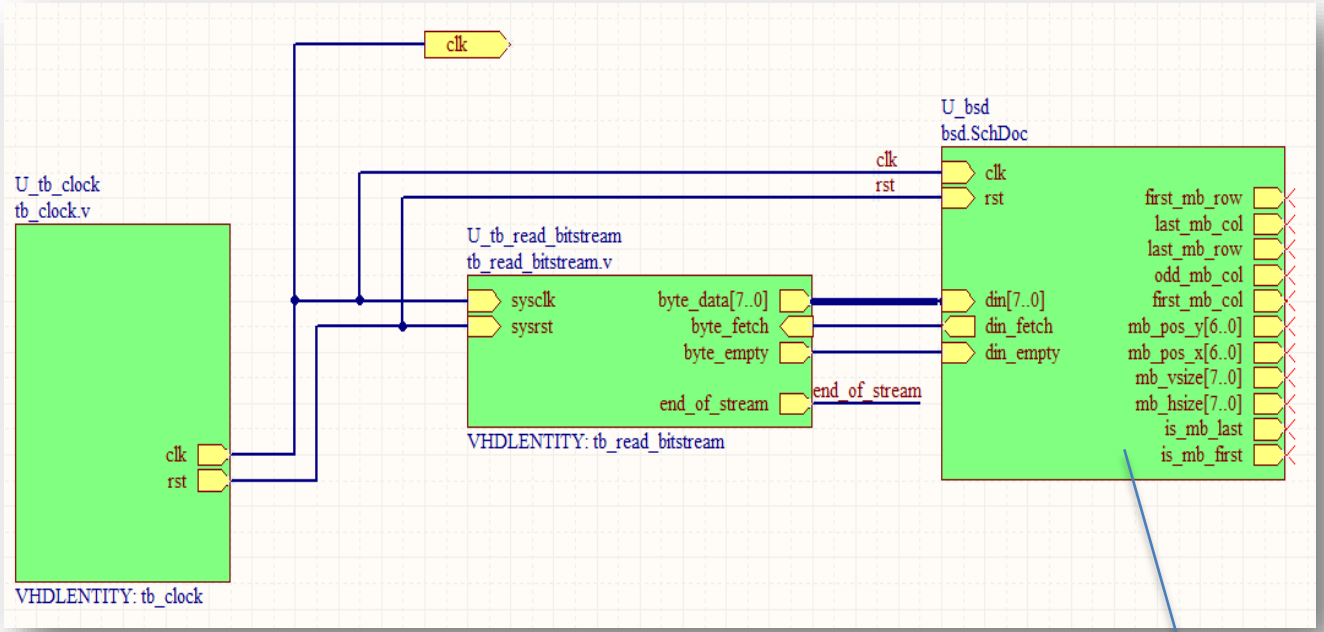
III.4 Development progress

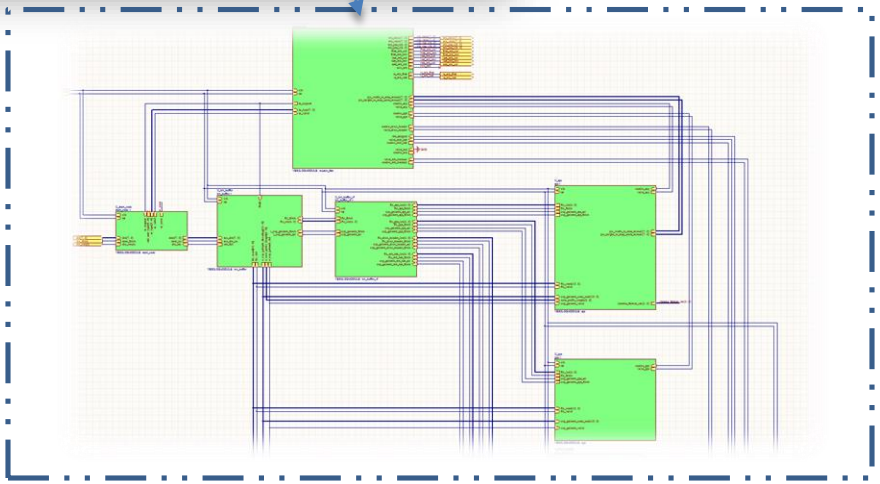


III.5 Tools used in the FPGA development flow design



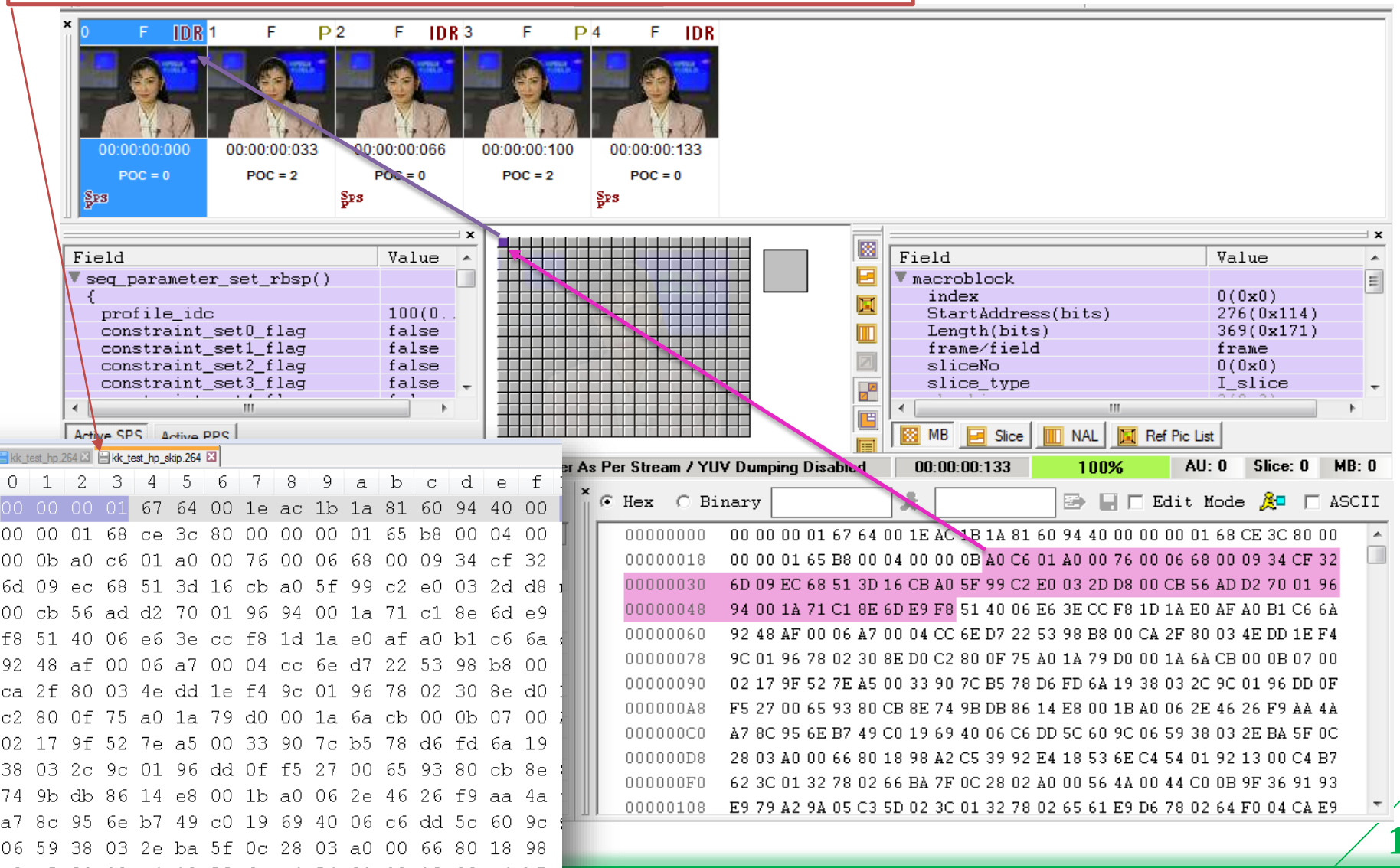
III.6 Simulation and results



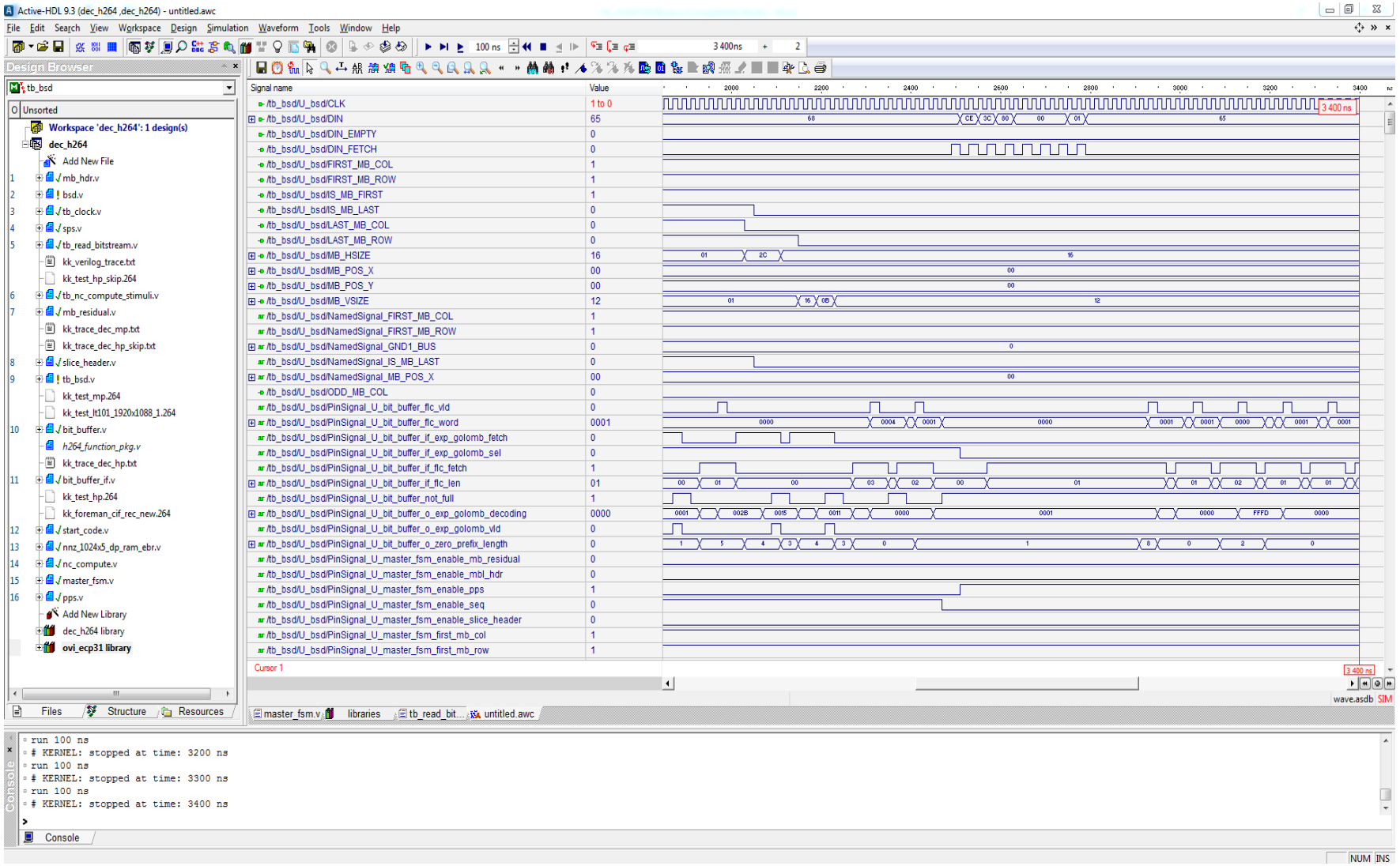


III.6 Simulation and results

Example of a video H.264 test file used in the simulations in ActiveHDL



III.6 Simulation and results



III.6 Simulation and results

Simulation results for SPS layer in Active HDL

/tb_bsd/U_bsd/U_sps/rst	0
/tb_bsd/U_bsd/U_sps/clk	1
/tb_bsd/U_bsd/U_sps/enable_seq	1
/tb_bsd/U_bsd/U_sps/r_state	12
/tb_bsd/U_bsd/U_sps/r_profile_idc	100
/tb_bsd/U_bsd/U_sps/r_constraint_set0_flag	0
/tb_bsd/U_bsd/U_sps/r_constraint_set1_flag	0
/tb_bsd/U_bsd/U_sps/r_constraint_set2_flag	0
/tb_bsd/U_bsd/U_sps/r_constraint_set3_flag	0
/tb_bsd/U_bsd/U_sps/r_reserved_zero_4bits	0
/tb_bsd/U_bsd/U_sps/r_level_idc	30
/tb_bsd/U_bsd/U_sps/r_seq_parameter_set_id_sps	0
/tb_bsd/U_bsd/U_sps/r_chroma_format_idc	1
/tb_bsd/U_bsd/U_sps/r_bit_depth_luma_minus8	0
/tb_bsd/U_bsd/U_sps/r_bit_depth_chroma_minus8	0
/tb_bsd/U_bsd/U_sps/r_qpprime_y_zero_transform_bypass_flag	0
/tb_bsd/U_bsd/U_sps/r_seq_scaling_matrix_present_flag	0
/tb_bsd/U_bsd/U_sps/r_log2_max_frame_num_minus4	12
/tb_bsd/U_bsd/U_sps/r_pic_order_cnt_type	0
/tb_bsd/U_bsd/U_sps/r_log2_max_pic_order_cnt_lsb_minus4	12
/tb_bsd/U_bsd/U_sps/r_num_ref_frames	1
/tb_bsd/U_bsd/U_sps/r_gaps_in_frame_num_value_allowed_flag	0
/tb_bsd/U_bsd/U_sps/r_pic_width_in_mbs_minus1	21
/tb_bsd/U_bsd/U_sps/r_pic_height_in_map_units_minus1	17
/tb_bsd/U_bsd/U_sps/r_frame_mbs_only_flag	1
/tb_bsd/U_bsd/U_sps/r_direct_8x8_inference_flag	0
/tb_bsd/U_bsd/U_sps/r_frame_cropping_flag	0
/tb_bsd/U_bsd/U_sps/r_vui_parameters_present_flag	0
/tb_bsd/U_bsd/U_sps/r_valid_seq	1

Trace file created by JM ITU-T C code

trace_dec.txt

```
1
2   SPS: profile_idc                                  01100100 (100)
3   SPS: constrained_set0_flag                               0 (  0)
4   SPS: constrained_set1_flag                               0 (  0)
5   SPS: constrained_set2_flag                               0 (  0)
6   SPS: constrained_set3_flag                               0 (  0)
7   SPS: reserved_zero_4bits                              0000 (  0)
8   SPS: level_idc                                    00011110 ( 30)
9   SPS: seq_parameter_set_id                                1 (  0)
10  SPS: chroma_format_idc                                 010 (  1)
11  SPS: bit_depth_luma_minus8                               1 (  0)
12  SPS: bit_depth_chroma_minus8                             1 (  0)
13  SPS: lossless_qpprime_y_zero_flag                        0 (  0)
14  SPS: seq_scaling_matrix_present_flag                     0 (  0)
15  SPS: log2_max_frame_num_minus4                     0001101 ( 12)
16  SPS: pic_order_cnt_type                                  1 (  0)
17  SPS: log2_max_pic_order_cnt_lsb_minus4             0001101 ( 12)
18  SPS: num_ref_frames                                    010 (  1)
19  SPS: gaps_in_frame_num_value_allowed_flag                0 (  0)
20  SPS: pic_width_in_mbs_minus1                       000010110 ( 21)
21  SPS: pic_height_in_map_units_minus1                000010010 ( 17)
22  SPS: frame_mbs_only_flag                                 1 (  1)
23  SPS: direct_8x8_inference_flag                           0 (  0)
24  SPS: frame_cropping_flag                                 0 (  0)
25  SPS: vui_parameters_present_flag                         0 (  0)
```

III.6 Simulation and results

Trace file created by AtiveHDL

Trace file created by JM ITU-T C code

kk_verilog_trace.txt - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

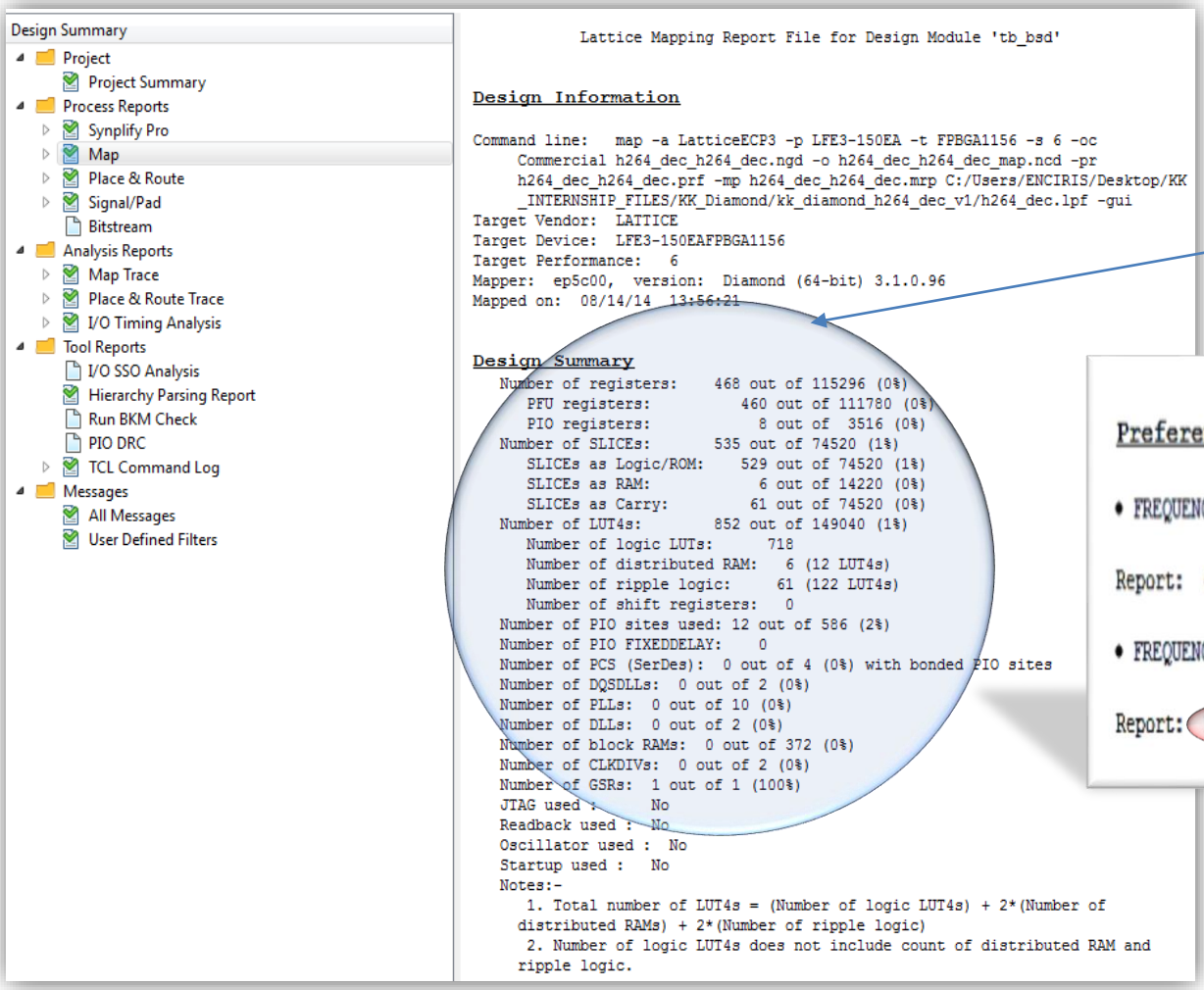
kk_verilog_trace.txt

```
SPS: profile_idc                                    01100100 (100)
SPS: constrained_set0_flag                                0 (  0)
SPS: constrained_set1_flag                                0 (  0)
SPS: constrained_set2_flag                                0 (  0)
SPS: constrained_set3_flag                                0 (  0)
SPS: reserved_zero_4bits                               0000 (  0)
SPS: level_idc                                      00011110 ( 30)
SPS: seq_parameter_set_id                                 1 (  0)
SPS: chroma_format_idc                                  010 (  1)
SPS: bit_depth_luma_minus8                                1 (  0)
SPS: bit_depth_chroma_minus8                              1 (  0)
SPS: lossless_qpprime_y_zero_flag                         0 (  0)
SPS: seq_scaling_matrix_present_flag                      0 (  0)
SPS: log2_max_frame_num_minus4                       0001101 ( 12)
SPS: pic_order_cnt_type                                   1 (  0)
SPS: log2_max_pic_order_cnt_lsb_minus4               0001101 ( 12)
SPS: num_ref_frames                                     010 (  1)
SPS: gaps_in_frame_num_value_allowed_flag                 0 (  0)
SPS: pic_width_in_mbs_minus1                         000010110 ( 21)
SPS: pic_height_in_map_units_minus1                  000010010 ( 17)
SPS: frame_mbs_only_flag                                  1 (  1)
SPS: direct_8x8_inference_flag                            0 (  0)
SPS: frame_cropping_flag                                  0 (  0)
SPS: vui_parameters_present_flag                          0 (  0)

PPS: pic_parameter_set_id                                 1 (  0)
PPS: seq_parameter_set_id                                 1 (  0)
PPS: entropy_coding_mode_flag                             0 (  0)
PPS: bottom_field_pic_order_in_frame_present_flag         0 (  0)
PPS: num_slice_groups_minus1                              1 (  0)
PPS: num_ref_idx_l0_default_active_minus1                 1 (  0)
PPS: num_ref_idx_l1_default_active_minus1                 1 (  0)
PPS: weighted_pred_flag                                   0 (  0)
PPS: weighted_bipred_idc                                 00 (  0)
PPS: pic_init_qp_minus26                                  1 (  0)
PPS: pic_init_qs_minus26                                  1 (  0)
PPS: chroma_qp_index_offset                               1 (  0)
PPS: deblocking_filter_control_present_flag               1 (  1)
PPS: constrained_intra_pred_flag                          0 (  0)
PPS: redundant_pic_cnt_present_flag                       0 (  0)

SH: first_mb_in_slice                                     1 (  0)
SH: slice_type                                          011 (  2)
SH: pic_parameter_set_id                                  1 (  0)
SH: frame_num                          0000000000000000 (  0)
SH: idr_pic_id                         0000000000000001 (  0)
SH: pic_order_cnt_lsb                  0000000000000000 (  0)
```

trace_dec.txt

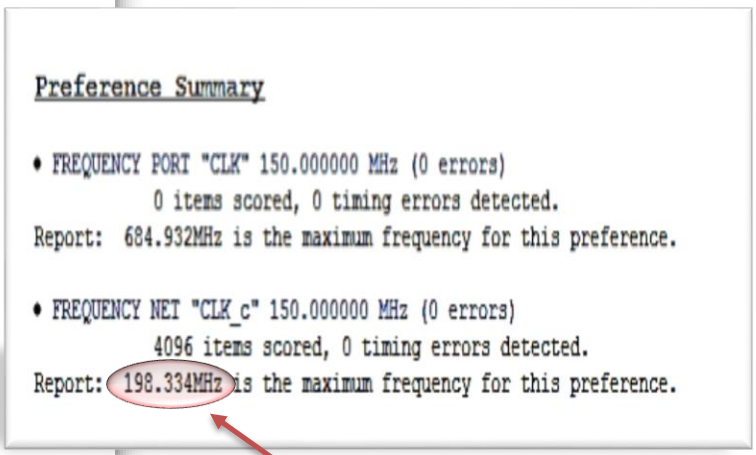
```
SPS: profile_idc                                    01100100 (100)
SPS: constrained_set0_flag                                0 (  0)
SPS: constrained_set1_flag                                0 (  0)
SPS: constrained_set2_flag                                0 (  0)
SPS: constrained_set3_flag                                0 (  0)
SPS: reserved_zero_4bits                               0000 (  0)
SPS: level_idc                                      00011110 ( 30)
SPS: seq_parameter_set_id                                 1 (  0)
SPS: chroma_format_idc                                  010 (  1)
SPS: bit_depth_luma_minus8                                1 (  0)
SPS: bit_depth_chroma_minus8                              1 (  0)
SPS: lossless_qpprime_y_zero_flag                         0 (  0)
SPS: seq_scaling_matrix_present_flag                      0 (  0)
SPS: log2_max_frame_num_minus4                       0001101 ( 12)
SPS: pic_order_cnt_type                                   1 (  0)
SPS: log2_max_pic_order_cnt_lsb_minus4               0001101 ( 12)
SPS: num_ref_frames                                     010 (  1)
SPS: gaps_in_frame_num_value_allowed_flag                 0 (  0)
SPS: pic_width_in_mbs_minus1                         000010110 ( 21)
SPS: pic_height_in_map_units_minus1                  000010010 ( 17)
SPS: frame_mbs_only_flag                                  1 (  1)
SPS: direct_8x8_inference_flag                            0 (  0)
SPS: frame_cropping_flag                                  0 (  0)
SPS: vui_parameters_present_flag                          0 (  0)

PPS: pic_parameter_set_id                                 1 (  0)
PPS: seq_parameter_set_id                                 1 (  0)
PPS: entropy_coding_mode_flag                             0 (  0)
PPS: bottom_field_pic_order_in_frame_present_flag         0 (  0)
PPS: num_slice_groups_minus1                              1 (  0)
PPS: num_ref_idx_l0_active_minus1                         1 (  0)
PPS: num_ref_idx_l1_active_minus1                         1 (  0)
PPS: weighted_pred_flag                                   0 (  0)
PPS: weighted_bipred_idc                                 00 (  0)
PPS: pic_init_qp_minus26                                  1 (  0)
PPS: pic_init_qs_minus26                                  1 (  0)
PPS: chroma_qp_index_offset                               1 (  0)
PPS: deblocking_filter_control_present_flag               1 (  1)
PPS: constrained_intra_pred_flag                          0 (  0)
PPS: redundant_pic_cnt_present_flag                       0 (  0)

SH: first_mb_in_slice                                     1 (  0)
SH: slice_type                                          011 (  2)
SH: pic_parameter_set_id                                  1 (  0)
SH: frame_num                          0000000000000000 (  0)
SH: idr_pic_id                                            1 (  0)
SH: pic_order_cnt_lsb                  0000000000000000 (  0)
```

III.6 Simulation and results



Lattice FPGA_LFE3-150EA-6FN1156C

Number of registers:	468 / 115296 (0 %)
Number of SLICEs:	535 / 74520 (1 %)
Number of LUT4s:	852 / 149040 (1 %)



Design can run on the maximum frequency of **198.334 MHz** with the clock constraint of 150 MHz

IV. Conclusion

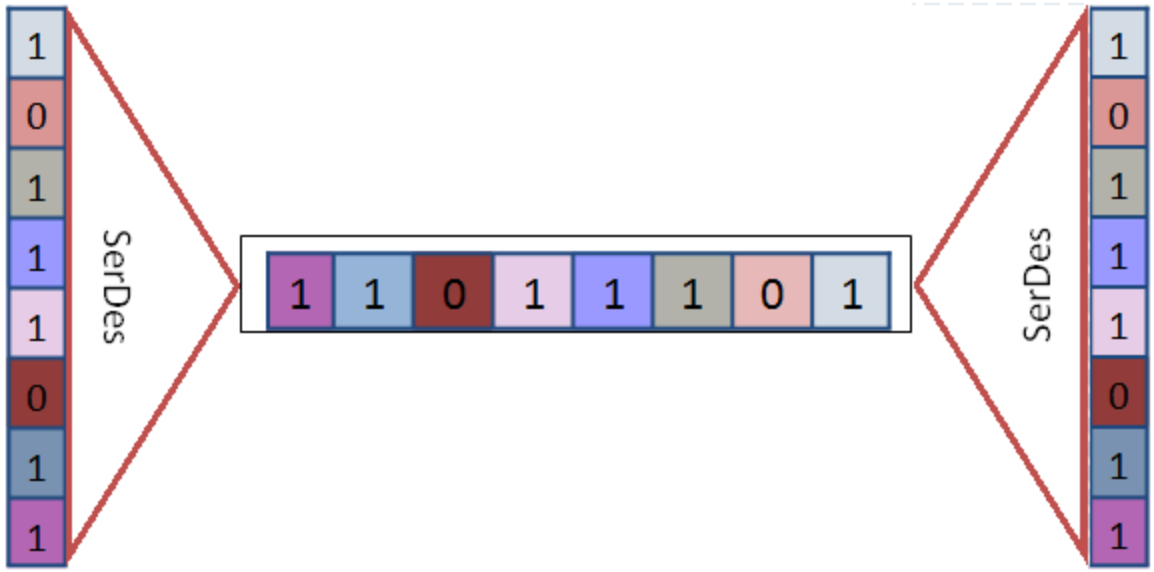
- Accomplished the first task on the implementation of a Lattice tri-rate SDI PHY IP core on the Enciris LT-125 board, provided an SMPTE 3G/HD/SD SDI video input functionality on the board.

- Because of lacking time and because of the complexity of CAVLD decoding, we can at the present manage to design the bitstream decoder which can decode the SPS, PPS, Slice header, and Macroblock header layer of the bitstream and which can run on the Lattice FPGA device - LFE3-150EA-6FN1156C with the maximum frequency of 198.334 MHz where the clock constraint is 150 MHz.

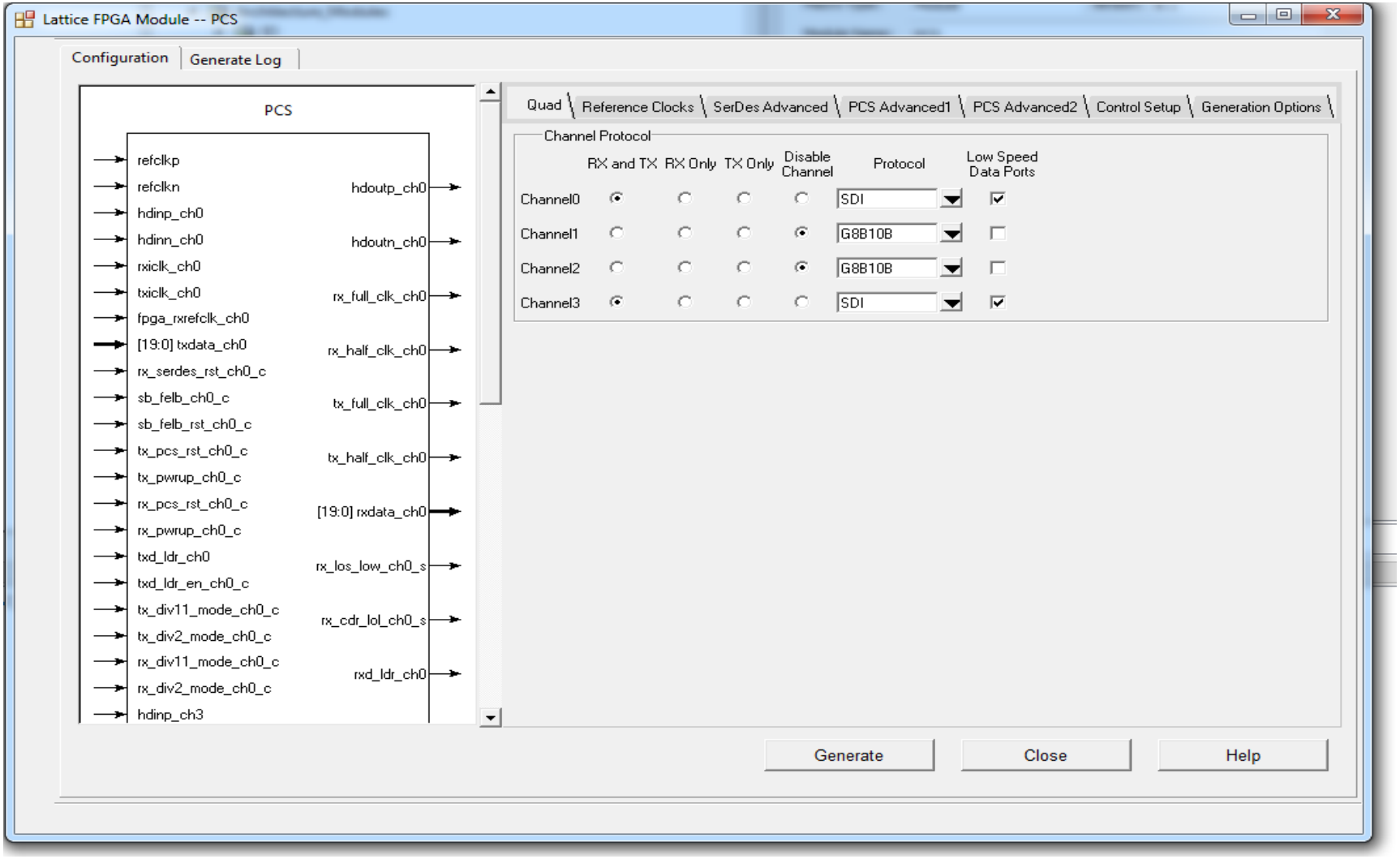
THANKS FOR YOUR ATTENTION

ANNEXE

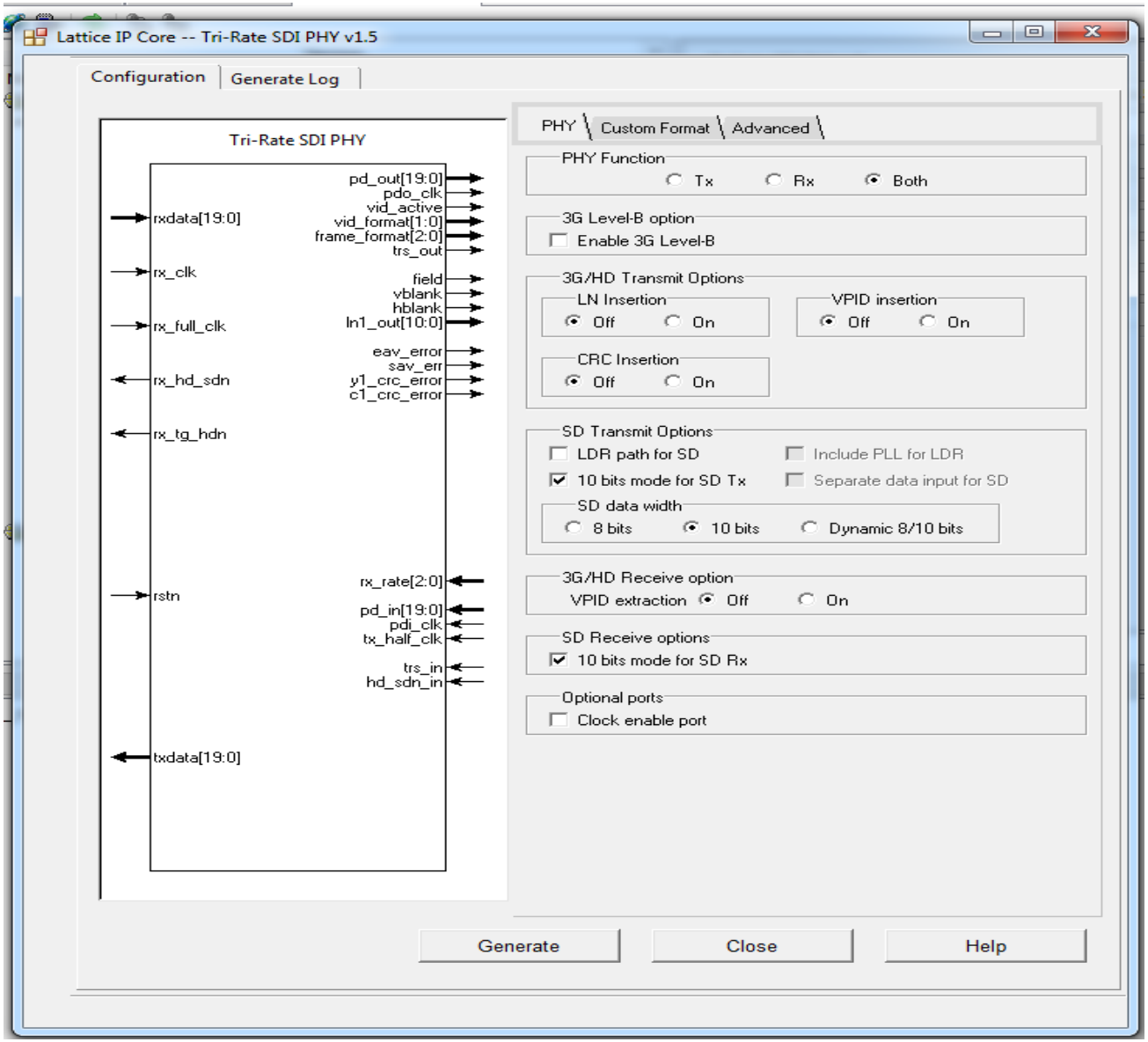
SERDES



IPexpress - SERDES

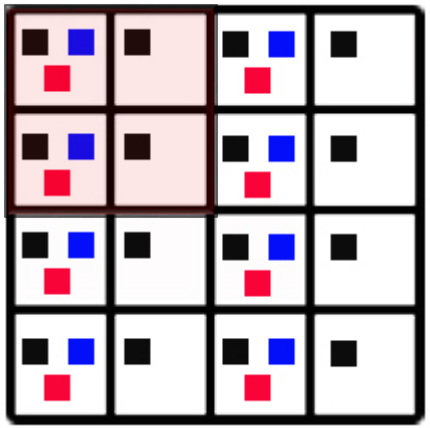


IPexpress - Tri-rate SDI IP Core

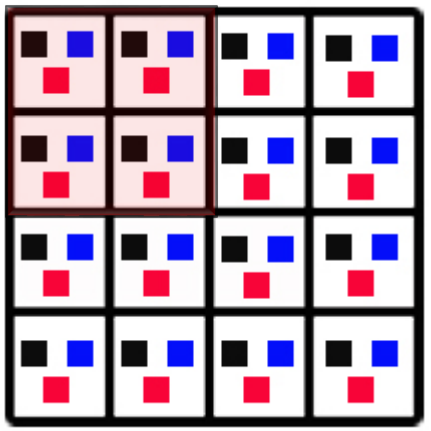


YC_bC_r 422 vs 444 sample

422



444

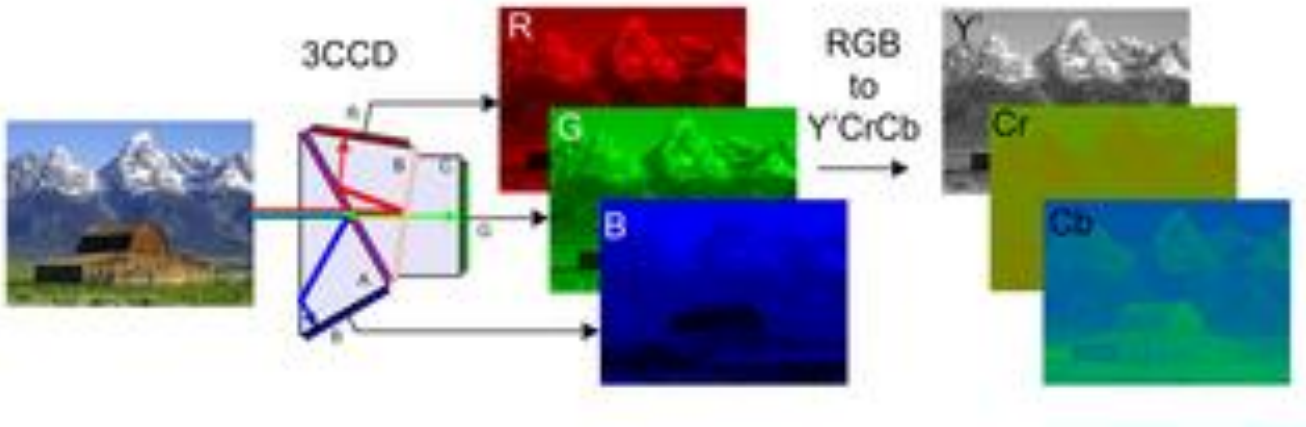


■ : Y

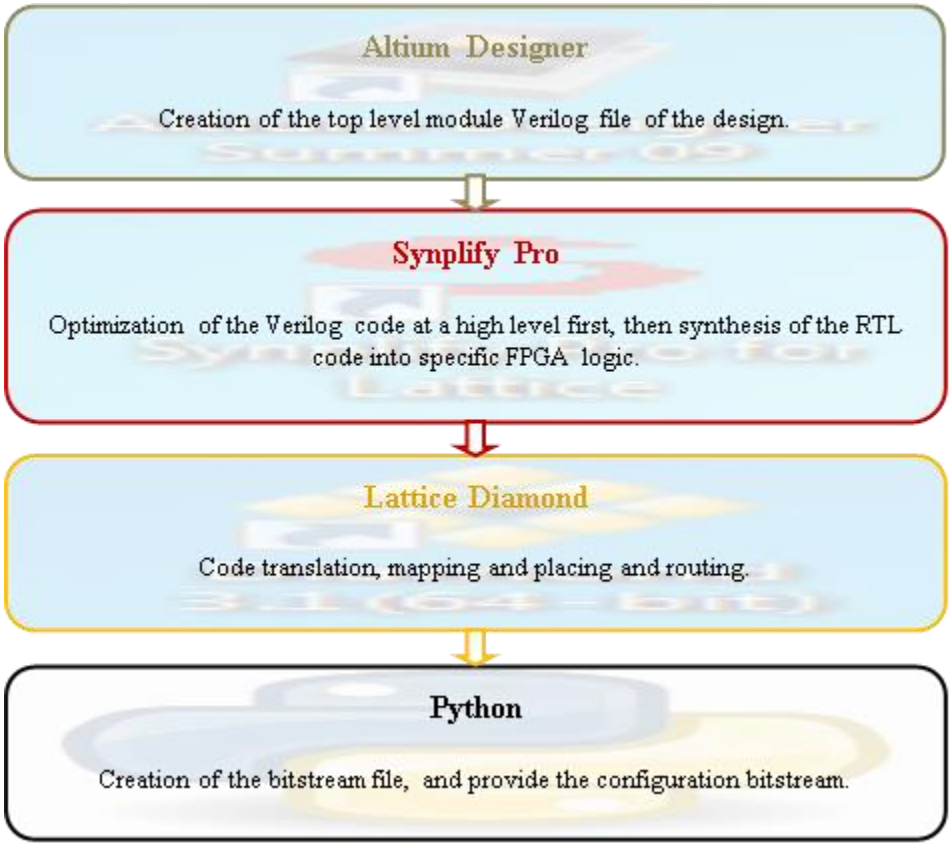
■ : C_b

■ : C_r

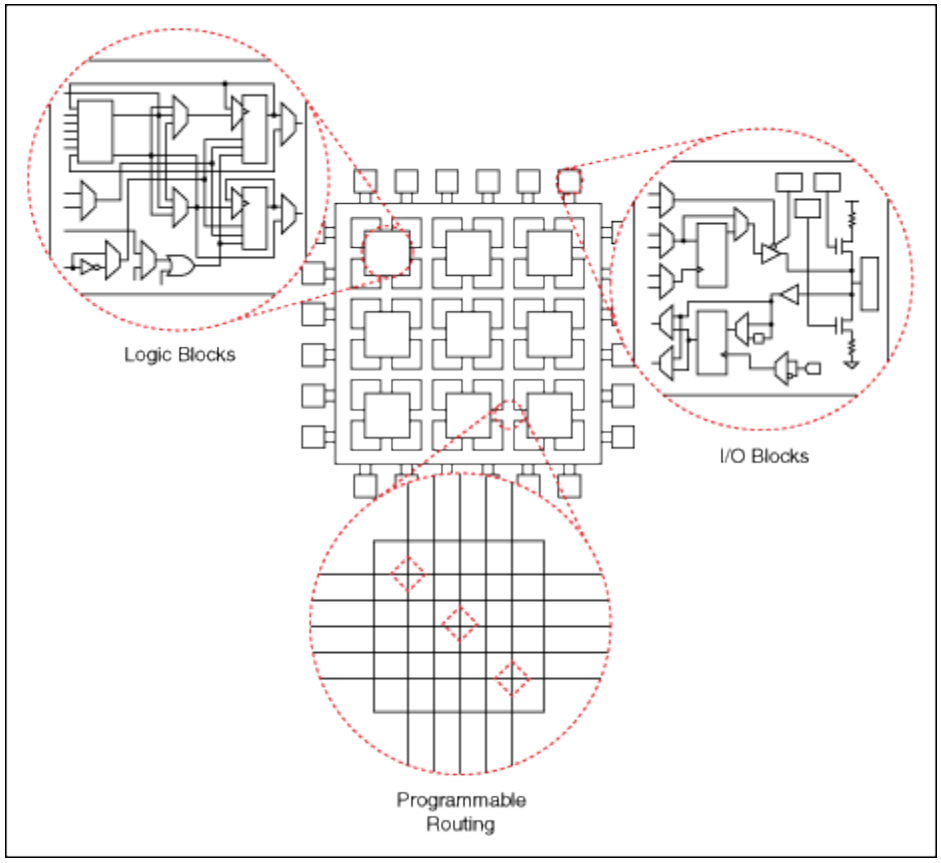
RGB to YC_bC_r



Flow of bitstream implementation



Introduction to FPGA Hardware Concepts (FPGA Module)



LUT

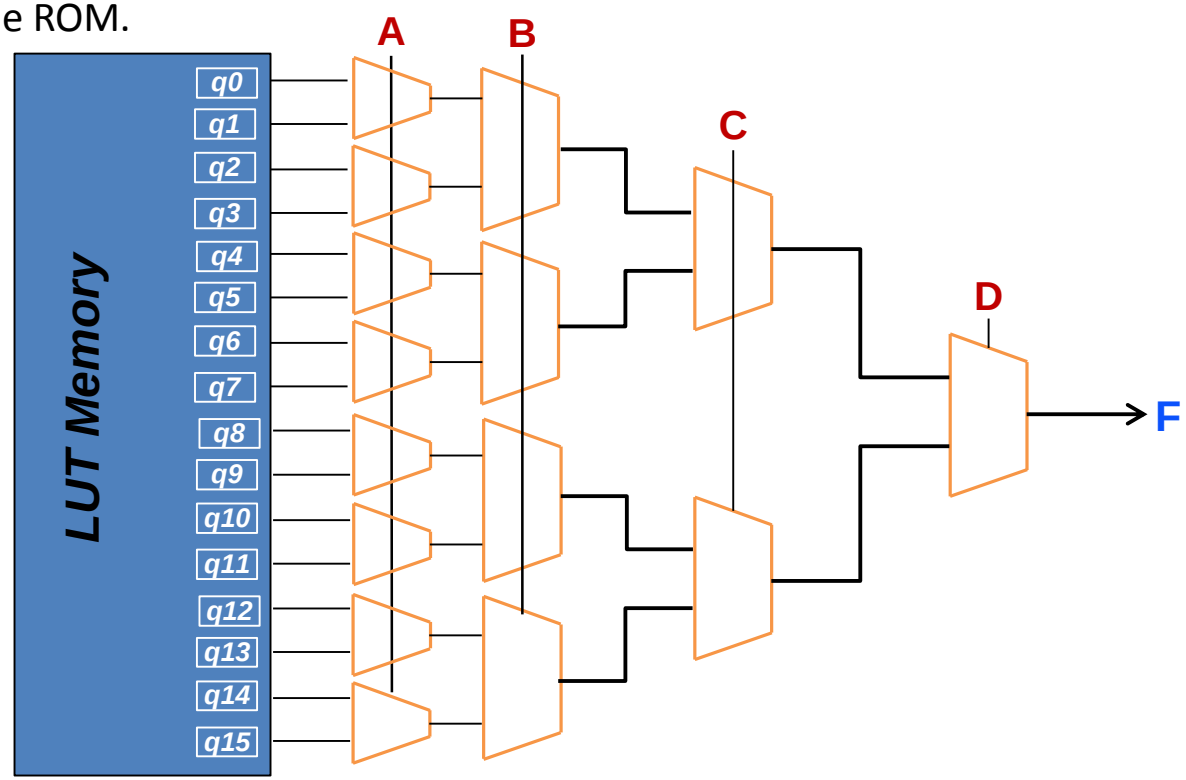
A lookup table (LUT) is used to transform the input data into a more desirable output format.

A lookup table (LUT) is a fast way to realize a complex function in digital logic. The address is the function input, and the value at that address is the function output. The advantage is that computing the function only takes a single memory lookup regardless of the complexity of the function, so is very fast. The disadvantage is that it takes memory, especially if you need high resolution for the function input.

For example, SIN is often implemented as a table lookup. If 10 bit angles are good enough resolution, then the whole function can be implemented as a lookup table with 1024 entries. (Actually in the case of SIN, only 1/4 cycle is stored then negated or indexed backwards depending on the actual quadrant, but that is a aside specific to SIN).
The function input can also be a combination of different input variables with the result expressed as a single integer. For example, a 4 x 4 bit multiply can be implemented as a lookup table of 256 values. The 8-bit table address can be the two 4-bit input values concatenated.

The look-up-table (LUT)

- Building the PFU from the inside out…
  - Nearly all FPGAs are based on a Look-Up-Table plus Register. Most are a LUT4. Aka LUT4+REG.
  - A 4-input LUT is just a 16-bit ROM, with 4 ‘address’ bits (ABCD) and a ‘data’ bit (F).
  - By programming the ROM, any 4 input logic functions can be formed.
    - Or it can be a simple ROM.



(‘q’ values are programmable SRAM memory bits that are determined through the design synthesis process)

Building ‘Slices’

Pairs of LUT+REG are grouped together with extra RAM/Ripple logic to form SLICEs.

Each Slice consists of:

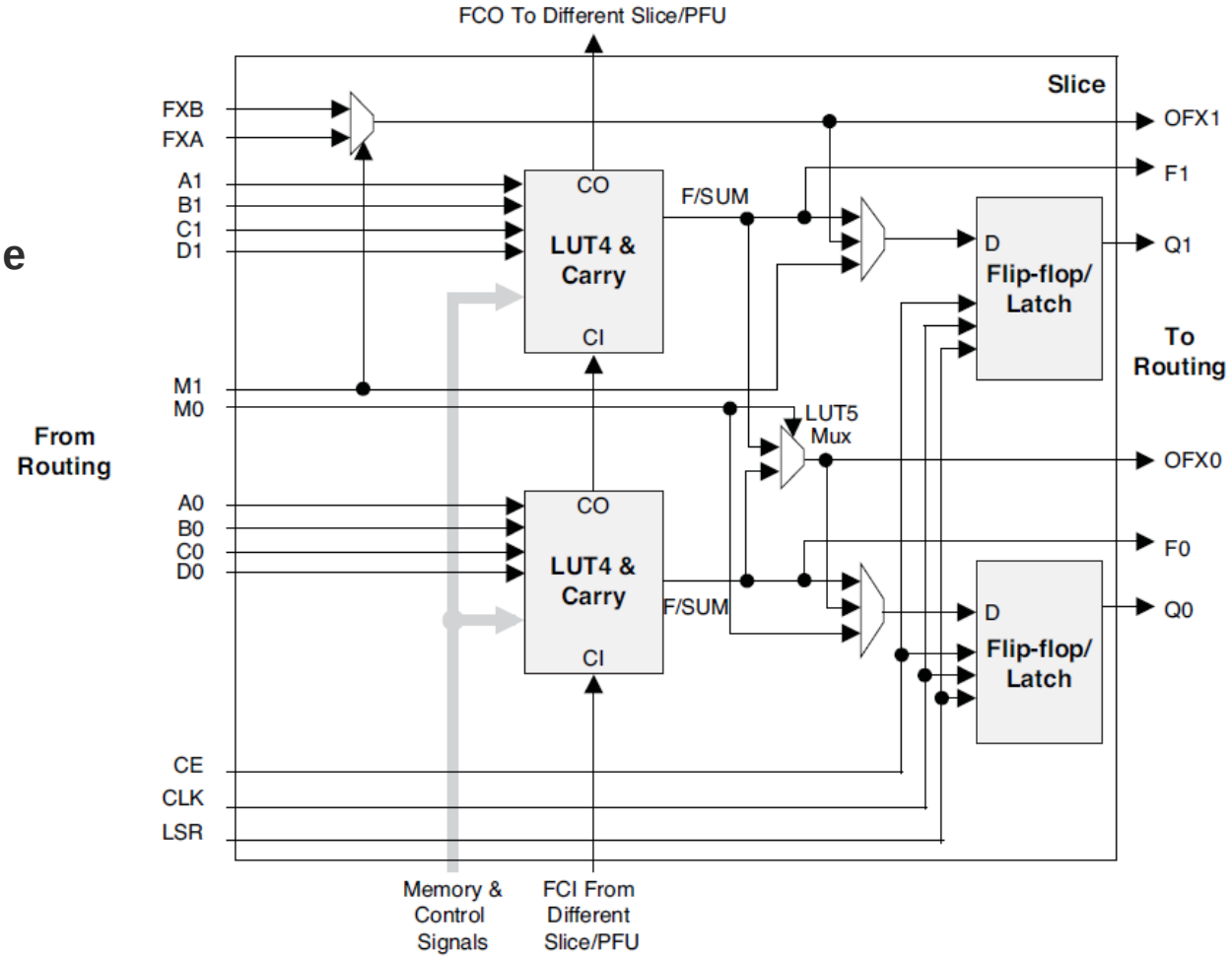
- Two 4-Input LUTs
- Two Registers
- Arithmetic Logic circuits
- Circuitry to support simple RAM mode

Slice Inputs:

- LUT Inputs: A, B, C, D
- Multi-Purpose Inputs: M
- Fast Carry Input: FCI
- Register Control Inputs: CLK, CE, LSR

Slice Outputs:

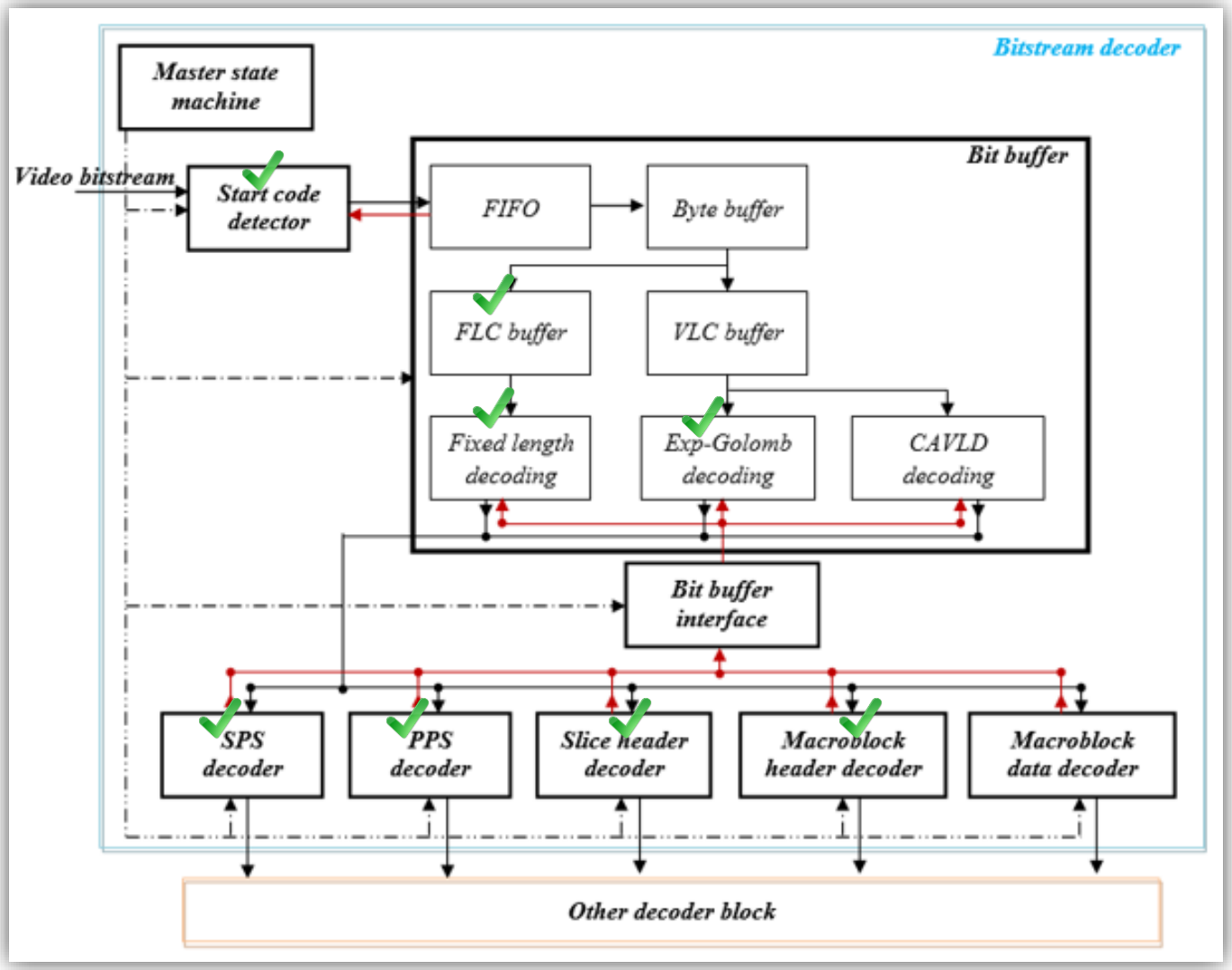
- LUT Outputs: F
- Register Outputs: Q
- Wide Function Outputs: OFX
- Fast Carry Output: FCO



For Slices 0 and 1, memory control signals are generated from Slice 2 as follows:

- WCK is CLK
- WRE is from LSR
- DI[3:2] for Slice 1 and DI[1:0] for Slice 0 data from Slice 2
- WAD [A:D] is a 4-bit address from slice 2 LUT input

H.264/AVC bitstream decoder hardware architecture



CAVLD decoding

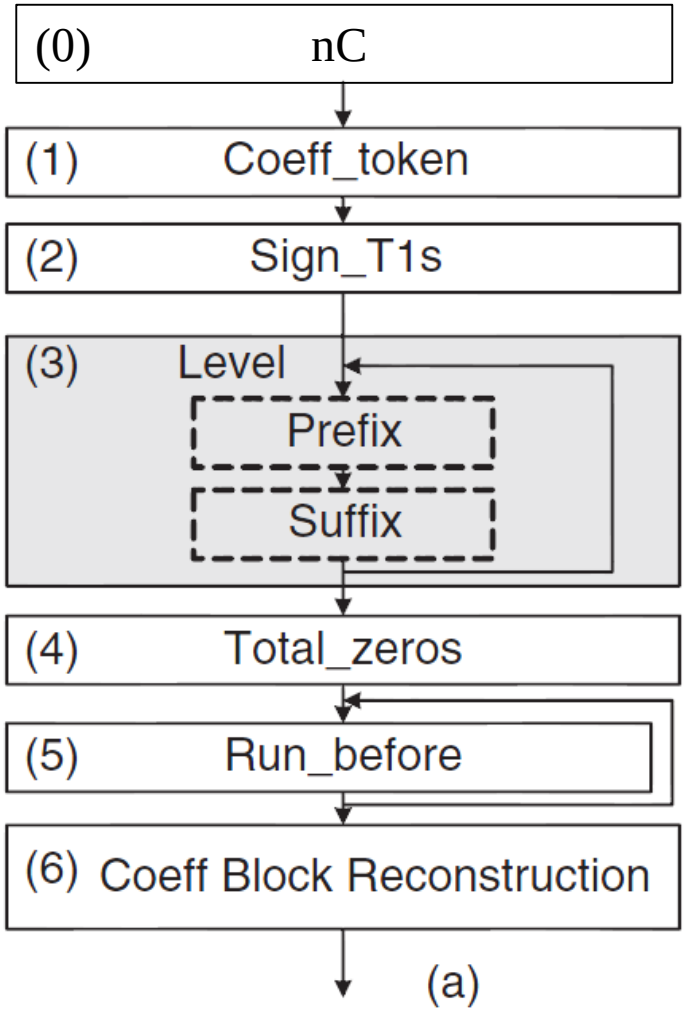


Table 9-5 – coeff_token mapping to TotalCoeff(coeff_token) and TrailingOnes(coeff_token)

TrailingOnes (coeff_token)	TotalCoeff (coeff_token)	0 <= nC < 2	2 <= nC < 4	4 <= nC < 8	8 <= nC	nC = = -1	nC = = -2
0	0	1	11	1111	0000 11	01	1
0	1	0001 01	0010 11	0011 11	0000 00	0001 11	0001 111
1	1	01	10	1110	0000 01	1	01
0	2	0000 0111	0001 11	0010 11	0001 00	0001 00	0001 110
1	2	0001 00	0011 1	0111 1	0001 01	0001 10	0001 101
2	2	001	011	1101	0001 10	001	001
0	3	0000 0011 1	0000 111	0010 00	0010 00	0000 11	0000 0011 1
1	3	0000 0110	0010 10	0110 0	0010 01	0000 011	0001 100
2	3	0000 101	0010 01	0111 0	0010 10	0000 010	0001 011
3	3	0001 1	0101	1100	0010 11	0001 01	0000 1
0	4	0000 0001 11	0000 0111	0001 111	0011 00	0000 10	0000 0011 0
1	4	0000 0011 0	0001 10	0101 0	0011 01	0000 0011	0000 0010 1
2	4	0000 0101	0001 01	0101 1	0011 10	0000 0010	0001 010
3	4	0000 11	0100	1011	0011 11	0000 000	0000 01

Node-Leaf method algorithm

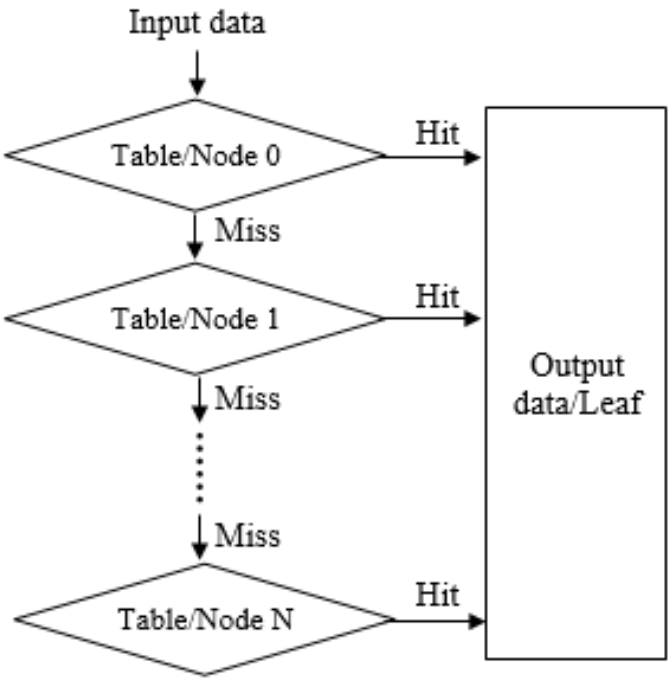


Table 9-7 : total_zeros tables for 4x4 blocks with TotalCoeff(coeff_token) 1 to 7

TotalCoeff(coeff_token) = 1

total_zeros	Len	Bin	Bin with don't cares (9 bits width)
15	9	000000001	000000001
14	9	000000010	000000010
13	9	000000011	000000011
12	8	00000010	00000010X
11	8	00000011	00000011X
10	7	0000010	0000010XX
9	7	0000011	0000011XX
8	6	000010	000010XXX
7	6	000011	000011XXX
6	5	00010	00010XXXX
5	5	00011	00011XXXX
4	4	0010	0010XXXXX
3	4	0011	0011XXXXX
2	3	010	010XXXXXX
1	3	011	011XXXXXX
0	1	1	1XXXXXXXX

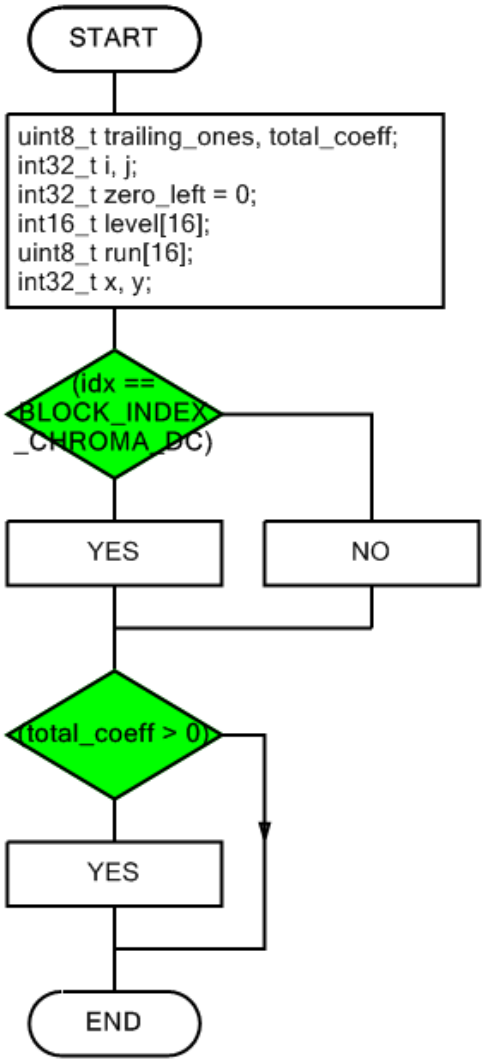
Number of table in that columns

5 x 8 entries

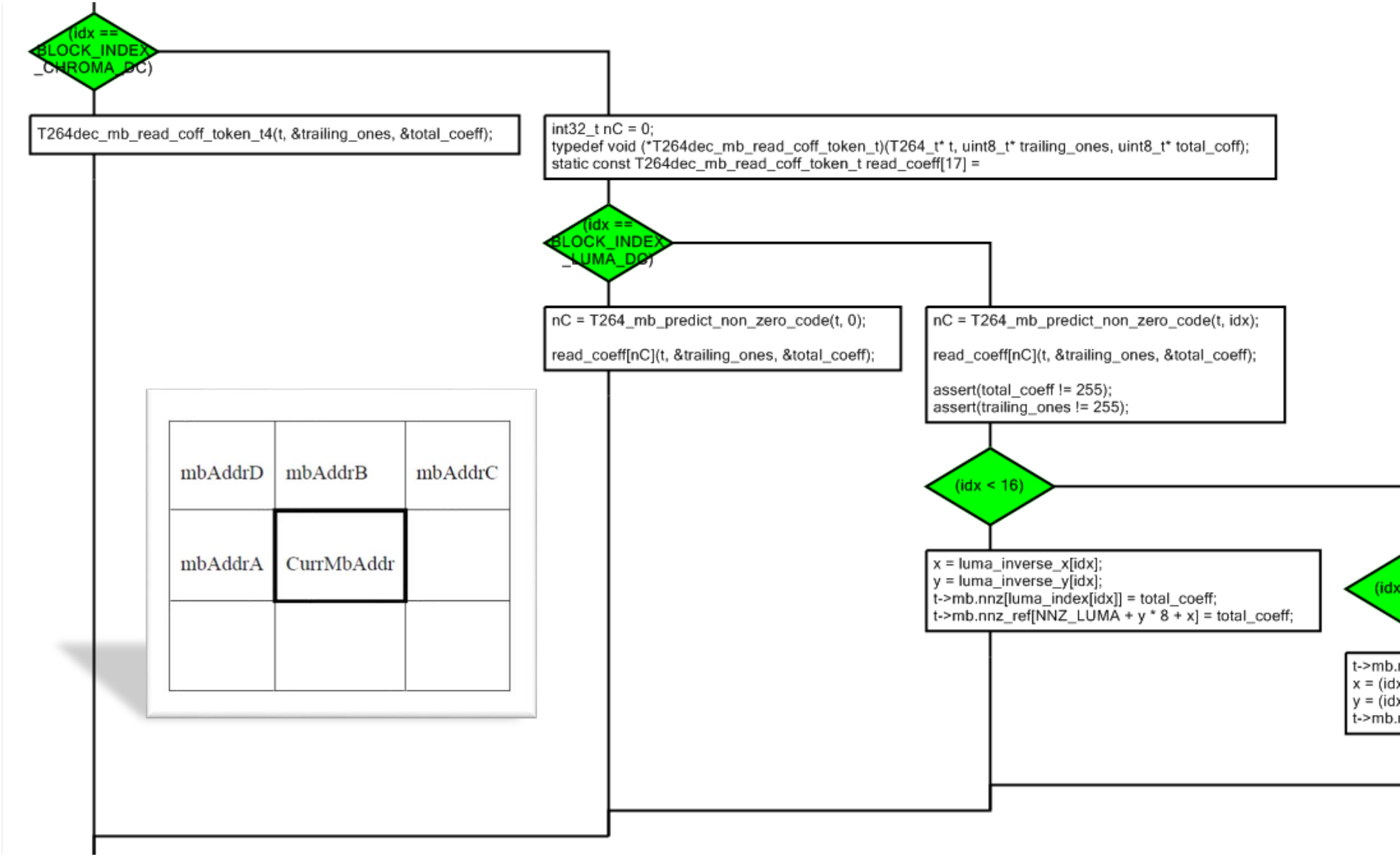
1	2	2
000	000	**001**
000	000	**010**
000	000	**011**
000	000	**10X**
000	000	**11X**
000	001	**0XX**
000	001	**1XX**
000	**010**	
000	**011**	
000	**10X**	
000	**11X**	
001	**0XX**	
001	**1XX**	
010		
011		
1XX		

Node 0
Node 1
Node 2
Node 3
Node 4

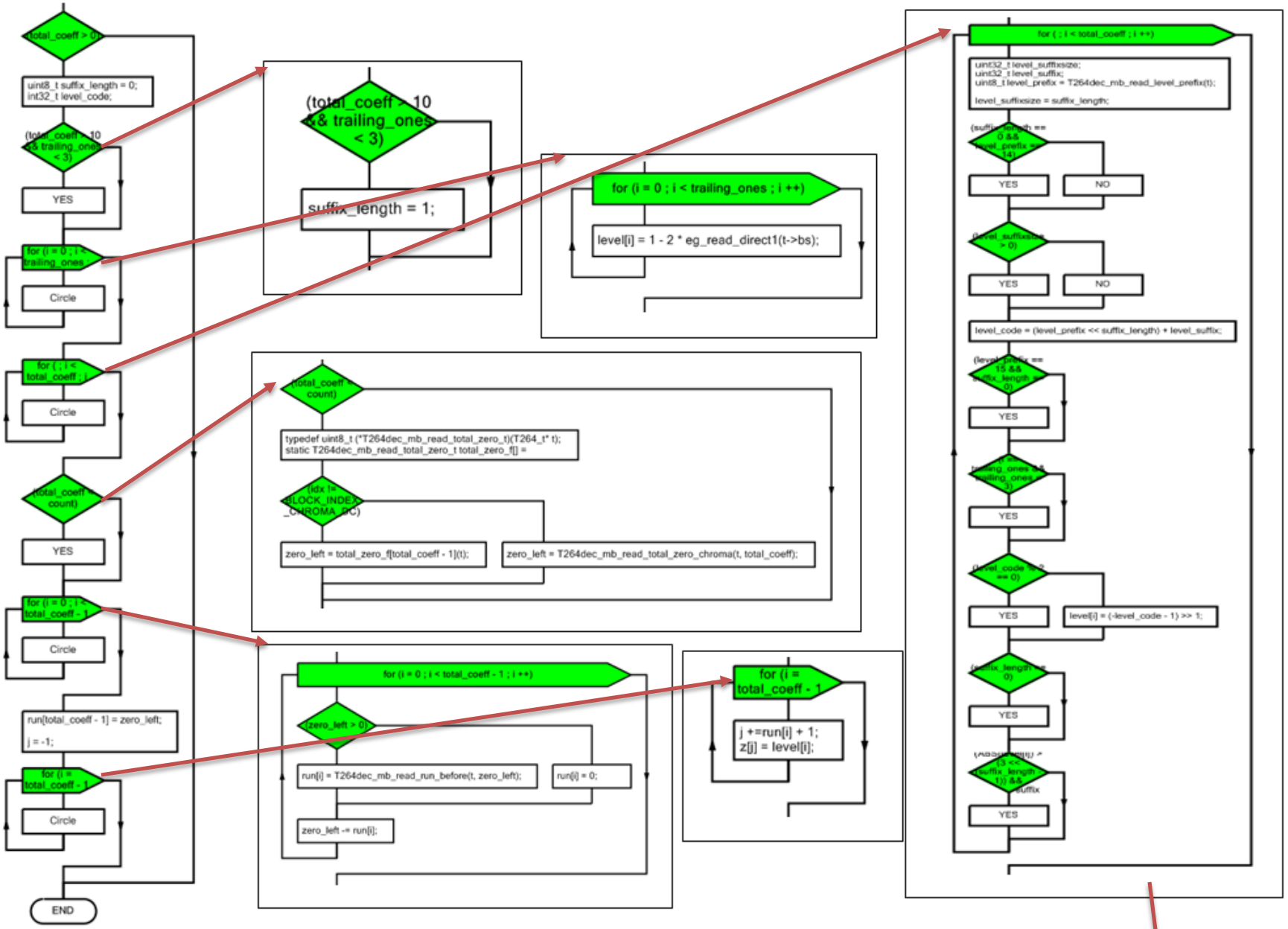
CAVLD decoding



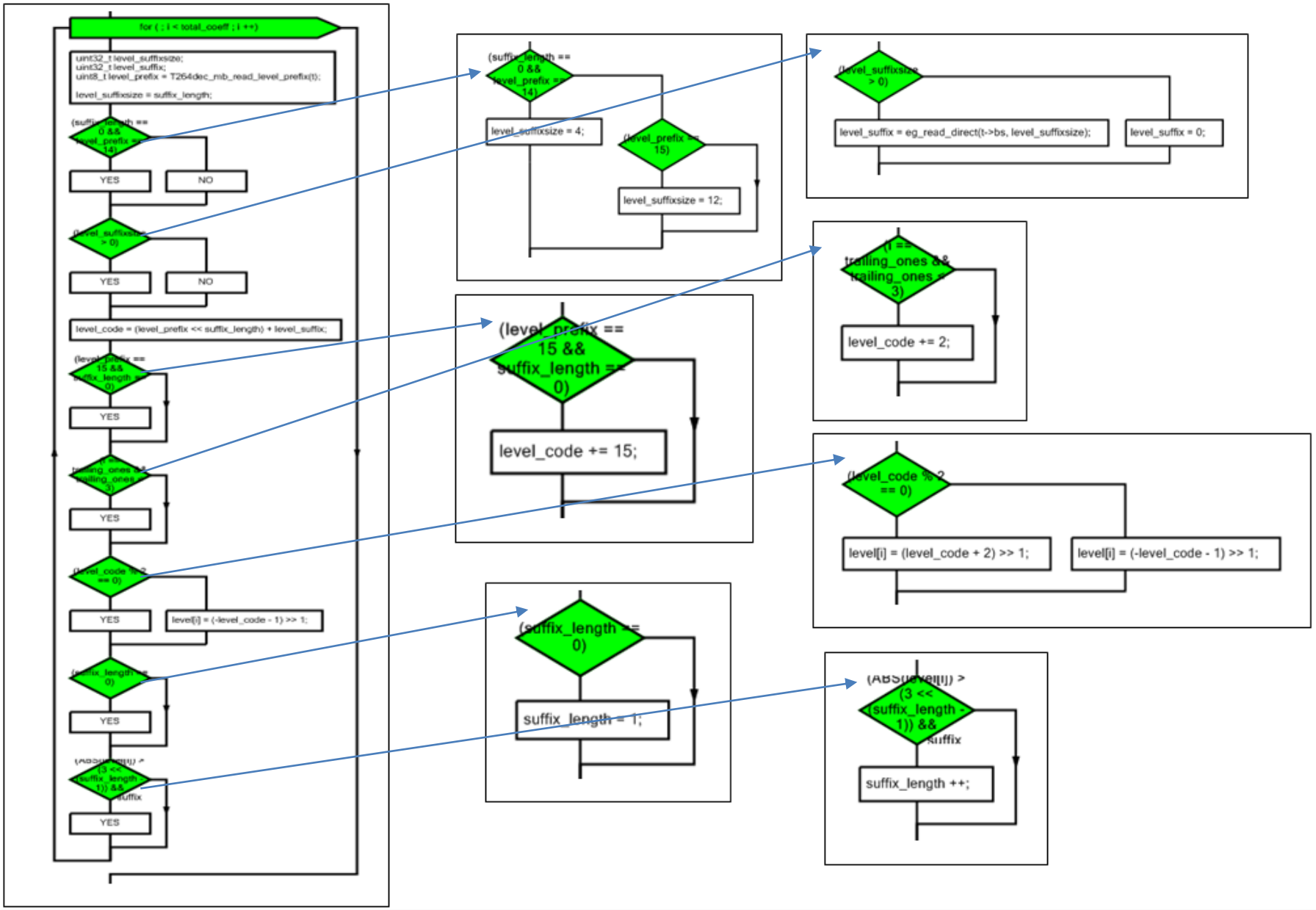
(0) nC


(idx == BLOCK_INDEX _CHROMA_DC)
T264dec_mb_read_coff_token_t4(t, &trailing_ones, &total_coeff);
int32_t nC = 0;
typedef void (*T264dec_mb_read_coff_token_t)(T264_t* t, uint8_t* trailing_ones, uint8_t* total_coff);
static const T264dec_mb_read_coff_token_t read_coeff[17] =
(idx == BLOCK_INDEX _LUMA_DC)
nC = T264_mb_predict_non_zero_code(t, 0);
read_coeff[nC](t, &trailing_ones, &total_coeff);
nC = T264_mb_predict_non_zero_code(t, idx);
read_coeff[nC](t, &trailing_ones, &total_coeff);
assert(total_coeff != 255);
assert(trailing_ones != 255);
(idx < 16)
x = luma_inverse_x[idx];
y = luma_inverse_y[idx];
t->mb.nnz[luma_index[idx]] = total_coeff;
t->mb.nnz_ref[NNZ_LUMA + y * 8 + x] = total_coeff;
mbAddrD
mbAddrB
mbAddrC
mbAddrA
CurrMbAddr

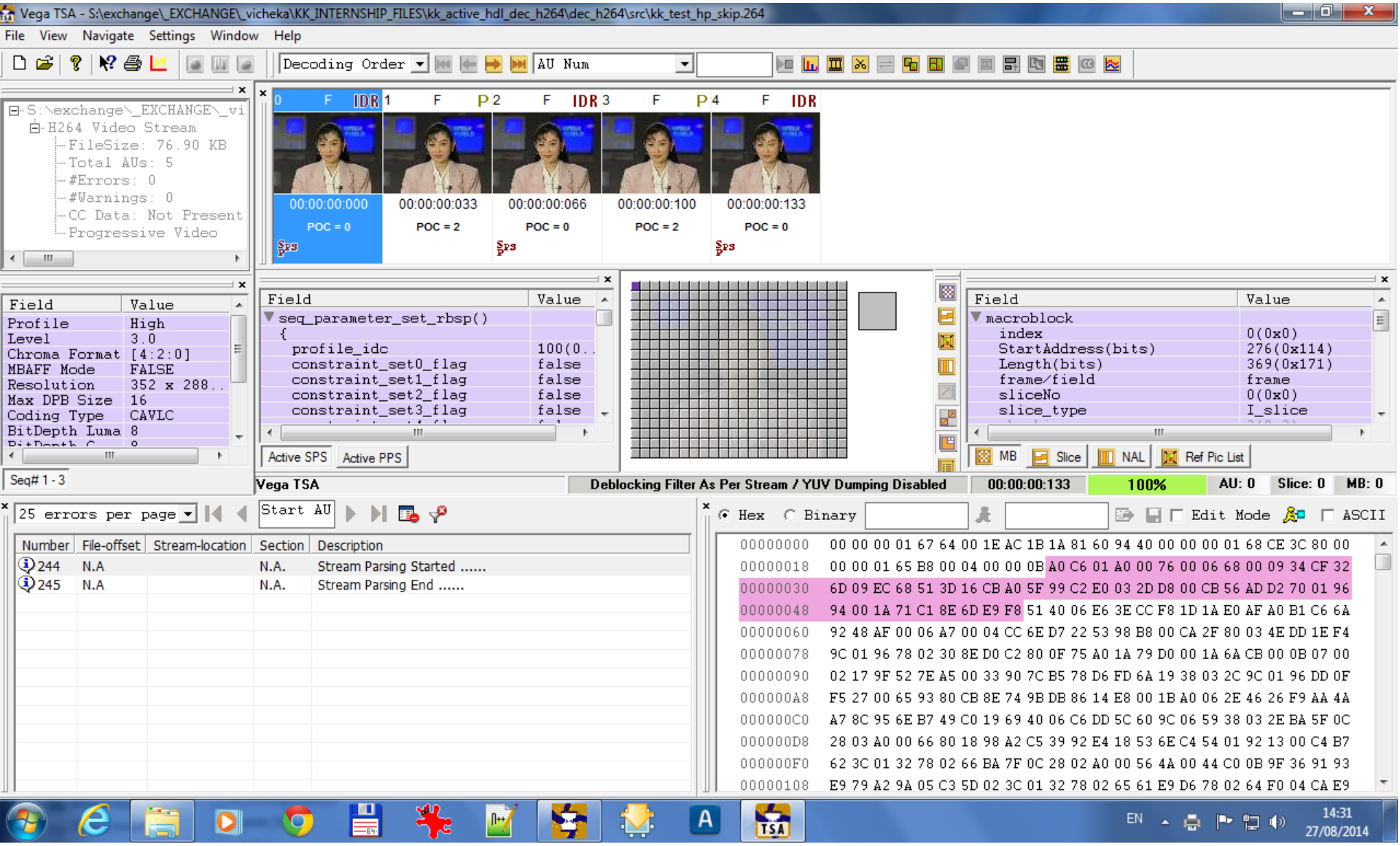
Step (1) to (6)


uint8_t suffix_length = 0;
int32_t level_code;
YES
for (i = 0 ; i < trailing_ones ; i ++)
Circle
for (; i < total_coeff ; i ++)
Circle
YES
for (i = 0 ; i < total_coeff - 1 ; i ++)
Circle
run[total_coeff - 1] = zero_left;
j = -1;
for (i = total_coeff - 1
Circle
END
(total_coeff > 10 && trailing_ones < 3)
suffix_length = 1;
for (i = 0 ; i < trailing_ones ; i ++)
level[i] = 1 - 2 * eg_read_direct1(t->bs);
typedef uint8_t (*T264dec_mb_read_total_zero_t)(T264_t* t);
static T264dec_mb_read_total_zero_t total_zero_f[] =
zero_left = total_zero_f[total_coeff - 1](t);
zero_left = T264dec_mb_read_total_zero_chroma(t, total_coeff);
for (i = 0 ; i < total_coeff - 1 ; i ++)
zero_left > 0)
run[i] = T264dec_mb_read_run_before(t, zero_left);
run[i] = 0;
zero_left -= run[i];
for (i = total_coeff - 1
j +=run[i] + 1;
z[j] = level[i];
for (; i < total_coeff ; i ++)
uint32_t level_suffixsize;
uint32_t level_suffix;
uint8_t level_prefix = T264dec_mb_read_level_prefix(t);
level_suffixsize = suffix_length;
YES
NO
level_code = (level_prefix << suffix_length) + level_suffix;
YES
YES
YES
level[i] = (-level_code - 1) >> 1;
YES
YES

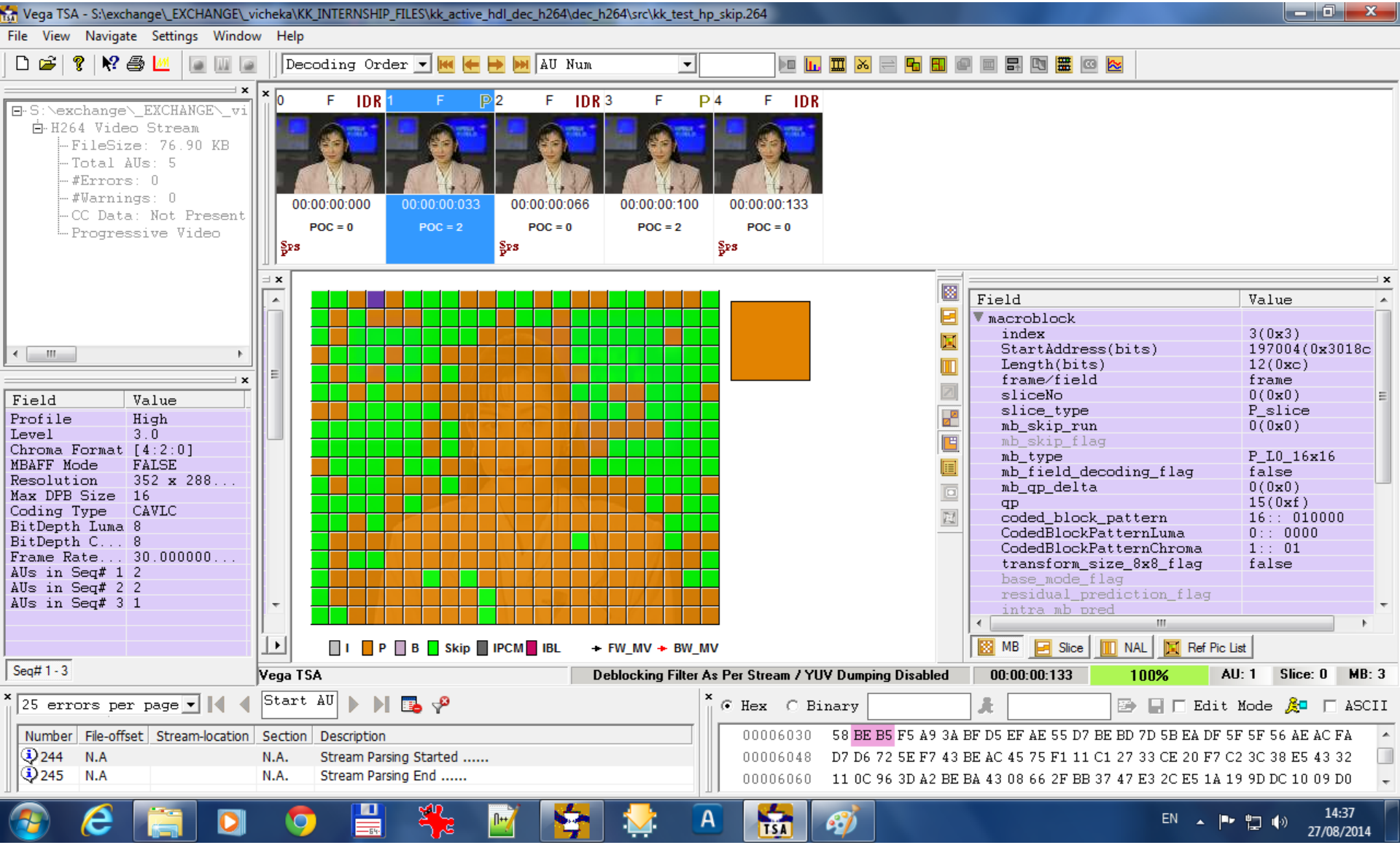
Step (1) to (6)


for (; i < total_coeff ; i ++)
uint32_t level_suffixsize;
uint32_t level_suffix;
uint8_t level_prefix = T264dec_mb_read_level_prefix(t);
level_suffixsize = suffix_length;
YES
NO
level_code = (level_prefix << suffix_length) + level_suffix;
level[i] = (-level_code - 1) >> 1;
level_suffixsize = 4;
level_suffixsize = 12;
level_suffix = eg_read_direct(t->bs, level_suffixsize);
level_suffix = 0;
level_code += 15;
level_code += 2;
level[i] = (level_code + 2) >> 1;
level[i] = (-level_code - 1) >> 1;
suffix_length = 1;
suffix_length ++;

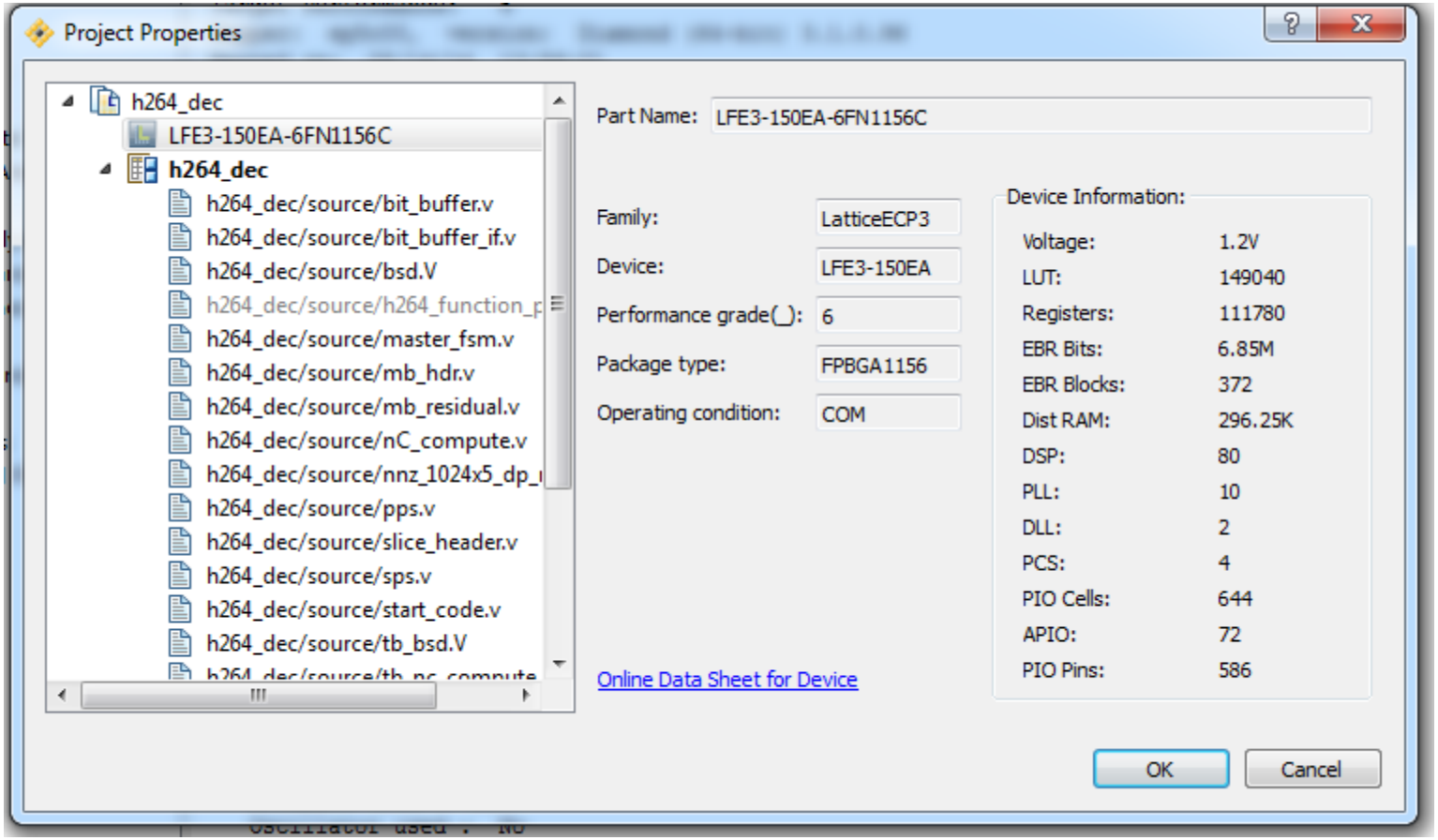
File .264 Analyse



File .264 Analyse



Lattice FPGA LFE3-150EA-6FN1156C



Simulation results of block SPS in ActiveHDL