Center for Embedded Computer Systems
University of California, Irvine

System Level Modeling of a H.264 Decoder

Weiwei Chen, Siwen Sun, Bin Zhang, Rainer Dömer

Technical Report CECS-08-10
August 12, 2008

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA
(949) 824-8059

{weiweic, s.sun, binz, doemer}@uci.edu
http://www.cecs.uci.edu/

System Level Modeling of a H.264 Decoder

Weiwei Chen, Siwen Sun, Bin Zhang, Rainer Dömer

Technical Report CECS-08-10
August 12, 2008

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA
(949) 824-8059

{weiweic, s.sun, binz, doemer}@uci.edu
http://www.cecs.uci.edu

Abstract

According to Moore's law, the number of transistors that can be effectively placed on an integrated circuit doubles every eighteen months. However, due to the current methodology, the productivity of IC design cannot increase correspondingly. To fill this gap, new methodologies which can increase the designer's productivity are needed. Modeling at higher levels of abstraction and the use of System Level Design Languages (SLDL), like SpecC or SystemC, are promising approaches to solve this problem.

In this report, we demonstrate the modeling of a H.264 decoder example of an industry-sized multi-media application. We first describe the essential features of the H.264 algorithm and its original reference implementation as sequential C code. We then recode the reference code into a system-level specification model with explicit representation of module hierarchy, connectivity, and testbench. Finally, we present a clean system model suitable for design space exploration and an initial platform model for implementation with integrated hardware and software components. All models of the H.264 decoder have been successfully validated against the initial reference C code.

Contents

List of Figures

List of Tables

System Level Modeling of a H.264 Decoder

Weiwei Chen, Siwen Sun, Bin Zhang, Rainer Dömer

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA

{weiweic, s.sun, binz, doemer}@uci.edu
http://www.cecs.uci.edu

Abstract

According to Moore's law, the number of transistors that can be effectively placed on an integrated circuit doubles every eighteen months. However, due to the current methodology, the productivity of IC design cannot increase correspondingly. To fill this gap, new methodologies which can increase the designer's productivity are needed. Modeling at higher levels of abstraction and the use of System Level Design Languages (SLDL), like SpecC or SystemC, are promising approaches to solve this problem.

In this report, we demonstrate the modeling of a H.264 decoder example of an industry-sized multi-media application. We first describe the essential features of the H.264 algorithm and its original reference implementation as sequential C code. We then recode the reference code into a system-level specification model with explicit representation of module hierarchy, connectivity, and testbench. Finally, we present a clean system model suitable for design space exploration and an initial platform model for implementation with integrated hardware and software components. All models of the H.264 decoder have been successfully validated against the initial reference C code.

1 Introduction

What is the essence of Design? Design is the process of model creation. At the starting point, the designer has two things on his hand: one is the idea (later formally called specification), another is the available resources (libraries are one example). The designer's job is to build a "bridge" to connect these two things, so later on, the manufacturer can "walk" on this "bridge" freely.

During this creative process, often a large mount of intermediate designs are created. How and when to create these intermediate designs depends on the design methodology. It is hard, or even impossible, to find the best. However, it is this freedom that defines the creativity of this process.

1.1 Electronic System Design

The Electronic System Designer has two things at his hands, the **application** and **transistors**. The application can be a mobile phone or an engine controller. The transistors can use 110nm technology or 32nm technology. The Electronic System Designer's job is to build a "bridge" by implementing the specification of the electronic system by use of transistors with specific technology.

According to Moore's Law: the number of transistors that can be effectively placed on an integrated circuit doubles every eighteen months. This means the number of combinations of these transistors increases even faster. Meanwhile, one product's time-to-market is decreasing due to fierce competition. It is a great challenge for the designer to explore these possible combinations in an efficient way.

However, some lessons can be learned from RTL design. When automated RTL design methods came out, it greatly increased the design productivity. The designer can map their applications by playing with adders, multiplexers or registers instead of AND gates or NOR gates. Now a higher level of abstraction, higher than RTL, would be a reasonable solution. By applying this so-called system level, the designer can start the complex systems from more abstract components, like processors memories and IPs. These higher abstractions, later on, can be refined down to RTL automatically or manually.

1.2 SpecC

SpecC [5] is an approach targeting this higher level of abstraction. Based on ANSI-C, additional features to facilitate the system level design are provided. With these features, the system designer can describe the desired system at a higher level than RTL. The details of the implementation can be ignored at the specification stage, and are refined later.

The features of SpecC that can facilitate system level design include:

- ***Data types***: event, signal, time, bit vectors
- ***Classes***: behavior, channel, interface
- ***Statements***: sequential, concurrent, pipelined, FSM

A detailed explanation of these features can be found in [4]

2 H.264 Decoder Algorithm

Before we model the H.264 example application at the system level using SpecC, we describe the background and some essential features of this algorithm.

2.1 H.264 Background

The H.264 and the MPEG-4 Part 10, also named Advanced Video Coding (AVC) standard, is jointly developed by ITU and ISO. It is a video compression standard with the intent to provide high quality video at a lower bit rate than previous standards. Another intent of H.264 is to satisfy a variety of video compression applications, from high bit rate to low bit rate, from high resolution to low resolution. These applications can be classified into 7 profiles [9]:

- Baseline Profile
- Main Profile
- Extended Profile
- High Profile
- High 10 Profile
- High 4:2:2 Profile
- High 4:4:4 Predictive Profile

An implementation of the H.264 decoder uses a .264 file as input. In this section, we will first go through the bit stream structure of a .264 file and then review the algorithm of H.264 decoder.

2.2 Brief Description of Input and Output Data Formats

H.264 is designed as a simple and straightforward video coding, which is able to completely decouple the transmission time, the decoding time, and the sampling of slices and pictures. A H.264 decoder takes a .264 file as input and outputs a raw YUV video stream. The decoding process specified in H.264 is unaware of time, and the H.264 syntax does not carry information such as the number of skipped frames [7].

2.2.1 H.264 Bitstream Formats

The bitstream of .264 file can be in either NAL unit stream format or byte stream format. The NAL unit stream format is conceptually the more basic type which is composed of a sequence of NAL units. Usually NAL units can be extracted from byte stream format by searching a unique prefix. An NAL unit specifies a generic format for use in both packet-oriented and bitstream systems. Figure 1 illustrates the structure of a basic NAL unit.

The elements of an NAL unit are as follows:

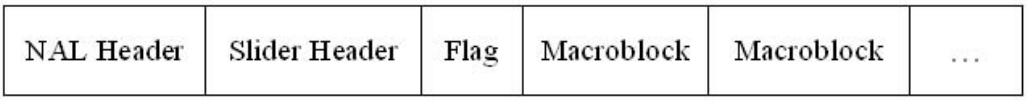


Figure 1: Structure of an NAL unit

- NAL Unit Header: 1 byte long, indicates the type of the bitstream and priority level
- Slice Header: contains authentication bits
- Flag: Start of the Data
- Macroblocks: contains data information; Each macroblock is comprised of one 16x16 luma array and, when the video format is not monochrome, two corresponding chroma sample arrays. When macroblock-adaptive frame/field decoding is not in use, each macroblock represents a spatial rectangular region of the picture.

2.2.2 YUV Bitstream Formats

The input of a H.264 encoder and output video of a H.264 decoder are both in .yuv formats. The source and decoded pictures (frames or fields) are each comprised of one or more sample arrays of chrominance and luminance. In ISO standard, the variables and terms associated with these arrays are referred to as luma (or L or Y) and chroma for convenience. The Y in YUV stands for luma, which is brightness, or lightness. U and V provide color information. In YUV video stream, there are three types of sampling for luma and chroma:

- 4:2:0 sampling: each of the two chroma arrays has half the height and half the width of the luma array.
- 4:2:2 sampling: each of the two chroma arrays has the same height and half the width of the luma array.
- 4:4:4 sampling: each of the two chroma arrays has the same height and width as the luma array.

Figure 2, Figure 3, and Figure 4 present the above three types of sampling in a frame, respectively.

2.3 H.264 Decoder Structure

The structure the of H.264 Decoder is depicted in Figure 5. The compressed video streams, from the network or other media, are firstly entropy decoded and reordered. Then the decoded data X will be de-quantized and de-transformed, which means the output data D'$_n$ are in time domain again. In the

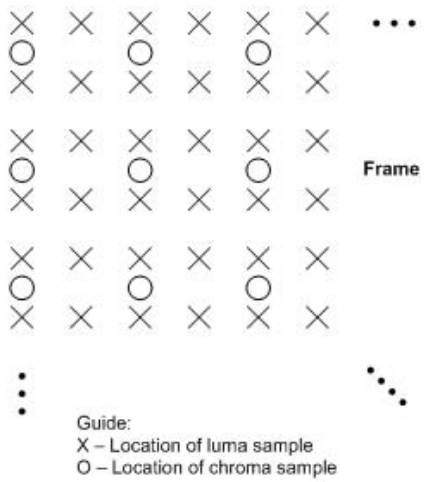


Figure 2: Format 4:2:0 luma and chroma samples in a frame [7]

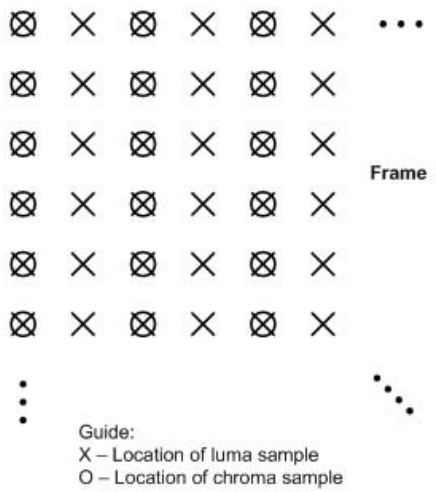


Figure 3: Format 4:2:2 luma and chroma samples in a frame [7]

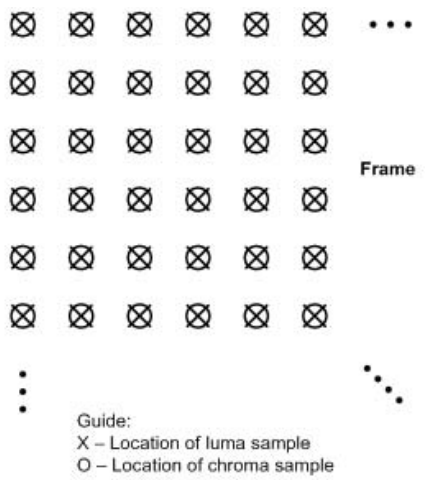


Figure 4: Format 4:4:4 luma and chroma samples in a frame [7]

final stage, the time domain data D'n are motion compensated, and the reconstructed video appears after the filter.

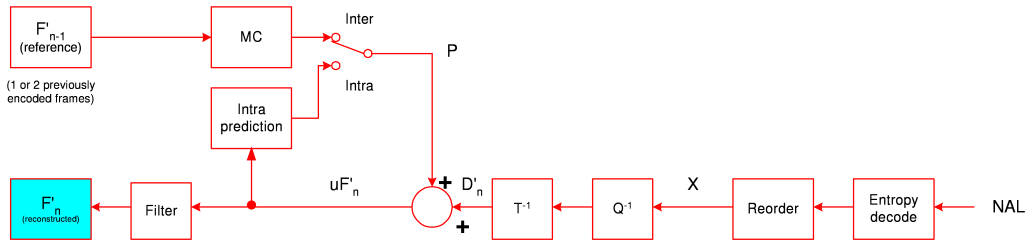


Figure 5: AVC decoder structure [8]

3 Reference C Implementation of H.264 Decoder

A specification model of a system is often not developed from scratch. Still it takes time for the designer to build the whole system specification. Today, embedded system designs are usually based on C reference source code. There are a great variety of C source codes for different algorithms from the free software organization on the internet. The embedded system designer can pick up those reference source codes, then recode them into SpecC SLDL for system exploration. Thus, designers can put more attention on system specification rather than writing every source code line for the algorithm.

3.1 Reference C Implementation of H.264 Decoder

SpecC SLDL is a superset of the C Language. It is compliant with ANSI-C which means that every ANSI-C program is SpecC program as well. It will be much easier to modify the source code in C than developing the specification from scratch.

For this project, we use the H.264/AVC JM Reference Software [6] as the reference implementation of our H.264 decoder specification model in SpecC. The H.264/AVC JM reference software is provided by the joint video team (JVT). The version we use is JM 13.0.

The H.264/AVC JM Reference Software consists of both the encoder (lencod) and the decoder (ldecod) for the H.264 standard. Our work focus on the decoder part. We will consider the encoder as an optional extension for the future.

Table 3.1 shows the properties of the reference source code in C.

Properties of the JM decoder reference C code	
Total number of the source files	33
Total number of the header files	36
Number of the source code lines	$\approx$ 30K
Number of functions	462

Table 1: Properties of the JM decoder reference source code

The reference source code is just a software decoder of H.264 standard without ensuring the realtime decoding ability. It can be compiled and run on Windows, Unix, MacOS and FreeBSD operating system platforms. Our work is done on Linux kernel 2.6.

3.2 JM H.264 Decoder Code Structure

The whole expansion of the decoder's function call tree of JM reference code is depicted in Figure 6. Comparing with the structure graph in the Figure 5, the **entropy decoding** block and **reordering** block are implemented in the function:

`read_one_macroblock()`
$\Leftarrow$ `decode_one_slice`
$\Leftarrow$ `decode_slice`

⇐ `decode_one_frame()`
⇐ `main()`.

The **de-quantizing** block, **de-transforming** block and **motion compensation** block are implemented in the function:
`decode_one_macroblock()`
⇐ `decode_one_slice`
⇐ `decode_slice`
⇐ `decode_one_frame()`
⇐ `main()`.

The detailed structure of the function `decode_one_macroblock()` is demonstrated in Figure 7. This function performs the key functionality of the H.264 standard, because in H.264 each frame is processed in units of macroblocks. For different types of macroblocks, which are indicated by `mb_type`, different transform functions are applied. These functions can be found in Figure 7 as well as listing 1.

```
if (IS_NEWINTRA (currMB))
{
  intrapred_luma_16x16();
  iMBtrans4x4();
  intra_cr_decoding();
}
else if (currMB->mb_type == I4MB)
{
  intrapred();
  intra_cr_decoding();
  itrans();
}
else if (currMB->mb_type == I8MB)
{
  intrapred8x8();
  intra_cr_decoding();
  itrans8x8();
}
else if ((img->type == P_SLICE) && (currMB->mb_type == PSKIP))
{
  perform_mc();
}
else if (currMB->mb_type == P16x16)
{
  iTransform();
  perform_mc();
}
else if (currMB->mb_type == P16x8)
{
  iTransform();
  perform_mc();
}
else if (currMB->mb_type == P8x16)
{
  iTransform();
  perform_mc();
}
else
{
  perform_mc();
}
```

Listing 1: Macro-block processing

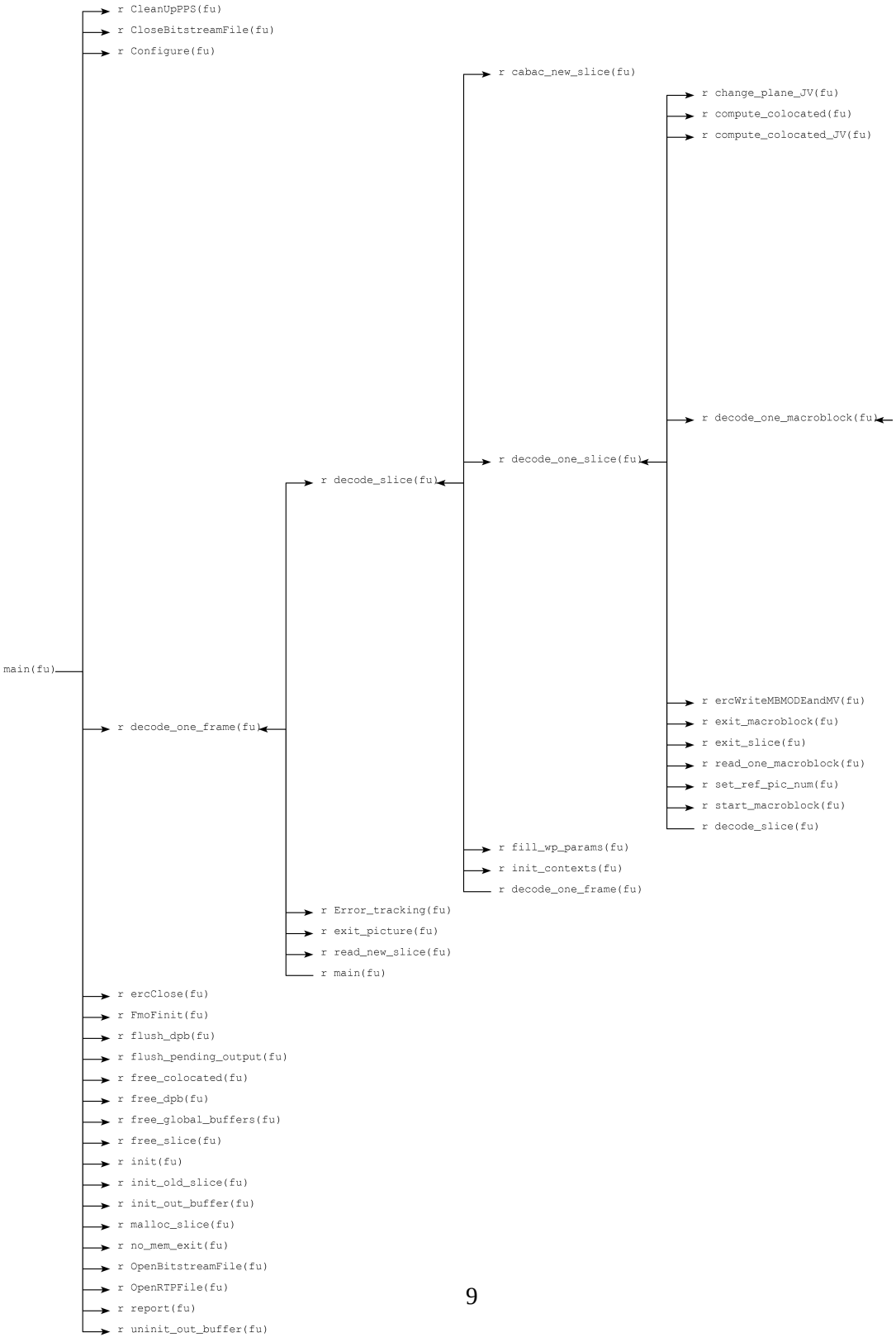


Figure 6: Main root expansion of the JM H.264 decoder function call tree

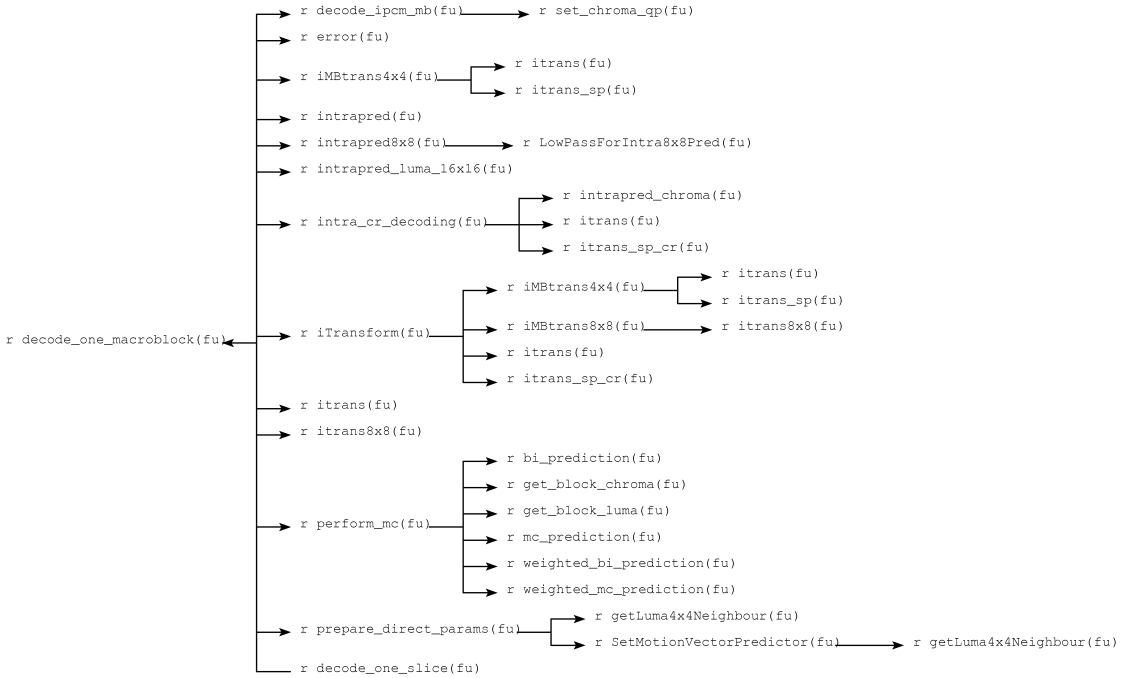


Figure 7: Expansion of decode_one_macroblock of the JM H.264 decoder function call tree

3.3 Make C Code SpecC compliant

The first step to built the specification model is to make the reference C code SpecC compliant. Since the reference code is not ANSI-C compliant, which is a feature restriction of the SpecC compiler, we have to make modifications to the reference code in order to make it pass the SpecC compilation.

Some of the modification issues are listed below:

- Local variable initialization: In SpecC, variable initializations at the time of declaration are restricted to constants. Some of the local variables in the functions are initialized as other former declared variables or variable addresses. Such local variables are manually modified by separating the declaration and initialized.

- Global variable initialization: Due to the same restriction, global variables which are initialized at the time of declaration are recoded for initialization and declaration separately. For global variables, the modification approach is different from that of local variables since we cannot assign the value outside any function scope. We create a function named *h264_scinit()* and put it at the beginning of the program for global variable initialization.

- Function pointer elimination: Function pointers can be used to simplify code, such as replace large switch statements. When dereferenced, a function will be invoked by passing zero or more arguments. However, SpecC does not support function pointers due to its ambiguity for high level synthesis. Modifications are necessary done to eliminate these pointers. When using the function pointer, we will first assign different function calls to it under different conditions. Then, when such a pointer is dereferenced, different functions can be invoked without considering the conditions. We eliminate function pointers by adding condition branches for different function calls instead of dereferencing function pointers. Figure 8 shows an example of how to eliminate function pointers in C.

- Macro definition conversion: In *ifunction.h*, the fundamental arithmetic operations are defined as macros. Some of them deal with the input parameters multiple times. However, whenever a parameter is passed as increments/decrements expression of a variable, the semantic of the source code will be changed in ANSI-C. Figure 9 shows the difference between macro definition and function call for certain arithmetic operation. Since macro definitions are not perfect, they are rewritten as function calls.

- Inline function elimination: SpecC does not support inline functions. *inline* is a reserved keyword of SpecC. Thus, the keywork *inline* is removed from inline function definitions.

- Source and header files renaming: The reference C source code includes 30 source files whose postfix is *.c*. We rename them with the same name and postfixed by *.sc* to identify them as SpecC source code. The reference C source code includes 36 header files whose postfix is *.h*. Since SpecC compiler will automatically generate header files for each SpecC source file with the same name postfixed by *.h*, we rename the original header file to be postfixed by *.sh*.

Source Code with Function Pointers

```
void (*fp) (void) = NULL;
int fp_cond;

void f1()
{
        printf("This is f1. \n");
}

void f2()
{
        printf("This is f2. \n");
}

void fp_init(int cond)
{
        switch(cond)
        {
        case 1:         fp = f1; break;
        case 2:         fp = f2; break;
        default:        fp = f3; break;
        }
}

void fp_call1()
{
        fp();
}

void fp_call2()
{
        fp();
}

int main()
{
        fp_cond = 1;
        fp_init(fp_cond);
        fp_call1();
        fp_call2();
}
```

Source Code without Function Pointers

```
// void (*fp) (void) = NULL;
int fp_cond;

void f1()
{
        printf("This is f1. \n");
}

void f2()
{
        printf("This is f2. \n");
}

void fp_call1()
{
        switch(fp_cond)
        {
        case 1:         f1(); break;
        case 2:         f2(); break;
        default:        f3(); break;
        }
}

void fp_call2()
{
        switch(fp_cond)
        {
        case 1:         f1(); break;
        case 2:         f2(); break;
        default:        f3(); break;
        }
}

int main()
{
        fp_cond = 1;
        fp_call1();
        fp_call2();
}
```

Figure 8: Example describing how to eliminate function pointers

Arithmetic Operation Macro

```
#define imin(a, b)     (((a) < (b)) ? (a) : (b))

void f()
{
   int a = 1, b = 2, x;
   x = imin(a++, b);     // x = 2, a = 3, b = 2;
}
```

Arithmetic Operation Function Call

```
int imin(int a, int b)
{
   return (((a) < (b)) ? (a) : (b));
}

void h()
{
   int a = 1, b = 2, x;
   x = fimin(a++, b);    // x = 1, a = 2, b = 2;
}
```

Figure 9: Example describing the difference between arithmetic operation macro and function call

- Header files ANSI-C compliant recoding: Three of the standard libraries, *huge_val.h, huge_valf.h, huge_vall.h* are not ANSI-C compliant due to the initialization of enumerated variables. We recoded them and converted them to user include files instead of system include files.

3.4 Simulation Result

After our compliance changes, the JM C reference code could be compiled and run on Linux kernel 2.6. The decoder inputs are:

- **test.264** The h.264 stream file. It can be a movie or a static picture.
- **decoder.cfg** The configuration file for the decoder. It provides the configuration information like the name of the input/output file, decoder rate, NAL mode, output chroma types, etc. The whole content of the test decoder configuration file is listed in Appendix A.1.
- **test_rec.yuv** The reference YUV file used to see whether the decoding process is correct or not by being compared with the output YUV file . This reference file is in YUV format.

The decoder output is:

- **test_dec.yuv** The output YUV file.

The decoder will display some information for each frame decoded, including frame number (*FRAME*), frame type, picture number (*POC*), quantization parameter (*QP*), signal to noise ratio of Y, U and V chroma components (*SnrY, SnrU, SnrV*), output chroma type (*YUV*), system decoding time (*time*), etc. If the decoding execution is correct, all the SNRs for Y, U and V components should be 0.

Appendix A.2 shows the simulation output of the JM reference implementation with our test file. It shows that the decoding execution is correct by those zero SNR values. The total number of frames in the test file is 299 and the system decoding time is 3.5 seconds real-time on the host machine (PC with Intel(R) Pentium(R) 4 CPU 3.00GHz).

4 H.264 Decoder Specification Model

The specification model in SLDL is the starting point for system level design including architecture exploration, communication refinement, profiling, performance evaluation, and so on. The specification captures all the functionality of the system but hides the implementation details. The specification model is an abstract model with the features for system level design and is based on the C reference implementation code. Since the specification model is the input of the next-step design exploration, a *good* specification model is very essential. References [3] and [2] talk about the important features of a good specification model:

- **Computation and Communication Separation:** In SpecC SLDL, computation units are modeled as behaviors, and communication units can be modeled as channels. Such separation allows easy plug-n-play module exploration in the specification model. It also is a key for efficient communication synthesis and design space exploration.
- **Modularity:** Modularity is required by the decomposion of the system functionality allowing to build the system hierarchically.
- **Granuality:** Granuality is determined by the size of the leaf behaviors. The higher the number of leaf behaviors, the greater the possibility for design space exploration. On the other hand, the smaller the granuality is the more complex the system specification will be. Proper granuality is a key feature of a good specification model.
- **Implementation details:** The specification model is an abstract system description which should not have any implicit and explicit implementation details. Abstracting away implementation details provides more feasibility for design space exploration in the later stage.
- **Concurrency:** Concurrency is the essence of hardware components. Any parallel functionality in the system should be considered to run concurrently for a more efficient architecture implementation.

4.1 H.264 Decoder Specification Model Creation

The SpecC specification model is built based on the JM C reference implementation. We mostly follow the hierarchy of the reference code and wrap behaviors according to their functionality. The guide rules for the specification model building are listed below:

- Separate those system calls, which are not synthesizable in hardware, like *malloc(), printf(), fopen(), fclose(), read(), write()*, from the main decoder algorithm.
- Create behaviors in top-down fashion according to the reference source code. The granularity of the behaviors is not smaller than those functions correspondingly represented in the H.264 algorithm block diagram.

Behaviors are built manually according to the rules above by inserting the C source code functions.

Our initial specification model consists of 37 behaviors. The top level includes three behaviors:

- **stimulus:** The module which provides the stimulus to the decoder. It parses the configuration file, opens the input stream file and the output YUV file, and deals with memory allocations. It prepares and feeds the data that the main decoder needs.

- **decoder:** The module which executes the H.264 decoding algorithm (design under test, DUT). It is the central part of our specification model. In this behavior, there are functional subbehaviors for H.264 decoding, including motion estimation, motion prediction, entropy decoding, slice and macroblock decoding, space transformation, etc.

- **monitor:** The module which monitors the status of the decoding process. This module it receives the decoding information of each frame and displays it at the same time when decoding the following frames. It will also receive the decoded data and store it into the output file.

These three behaviors build up the testbench of our specification model. Figure 10 shows the structure of this testbench. Figure 11 shows the whole spanned behavioral hierarchy tree.

4.2 Simulation Result

We compiled our H.264 decoder specification model on Linux kernel 2.6 by SpecC Compiler (SCC) V.2.2.1 . The specification model could be compiled into an executable program and runs well on Linux 2.6.

Appendix A.3 shows the simulation output of the H.264 specification model with our test file. It shows that the decoding execution is correct by those zero SNR values. The total number of frames in the test file is 299 and the system decoding time is 4.9 seconds real-time on the host machine (PC with Intel(R) Pentium(R) 4 CPU 3.00GHz).

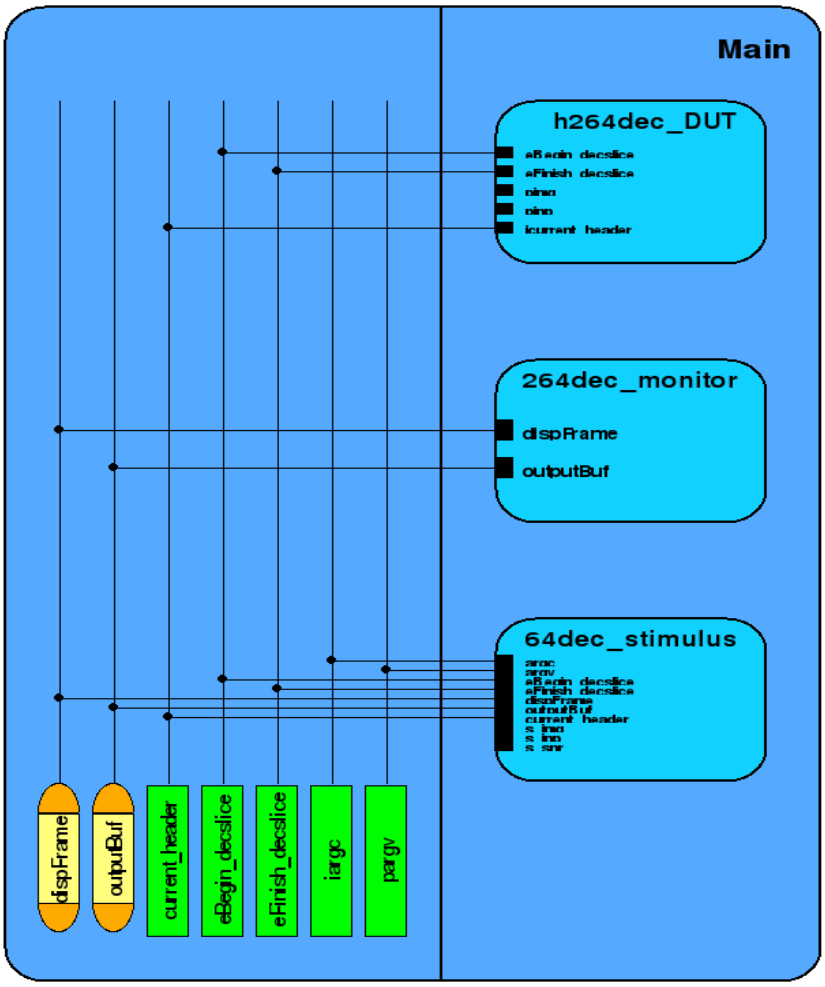


Figure 10: Top level testbench of H.264 decoder specification model

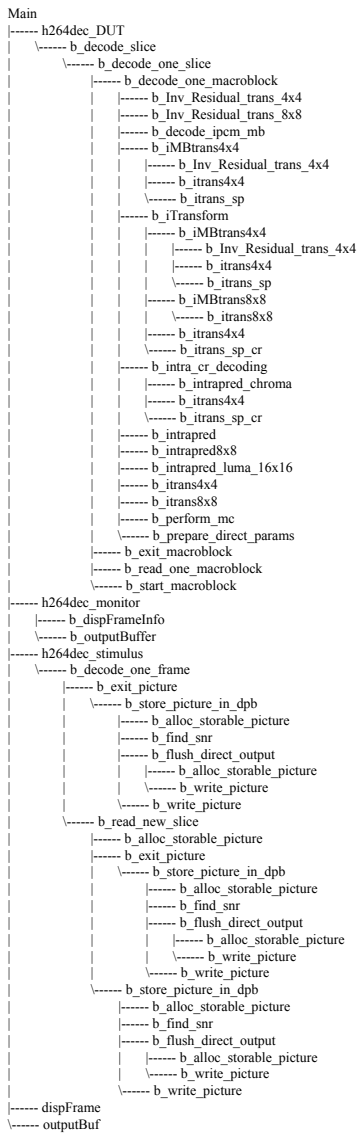


Figure 11: Behavioral hierarchy tree of H.264 decoder specification model

5 H.264 Decoder Exploration Model

5.1 Exploration Model

The H.264 specification model is built for system level design exploration. We use System-on-chip Environment (SCE) for further modeling, analysis, synthesis and validation. SCE is a tool with a graphical user interface (GUI) and a set of tools which facilitate the system level design flow [1]. The tool set includes a profiling tool, several refinement tools, estimation tools, and so on.

A clean specification model is needed for SCE to do the further exploration work. A clean specification model is the one in which only the leaf behaviors contain the C code and all the child behaviors are composed either in *parallel* (using *par* statement), or in *pipeline* (using *pipe* statement), or in *Finite State Machine (FSM)* style (using *fsm* statement), or sequentially [2]. However, the specification in Section 4 is not yet a clean exploration model.

We will discuss how to build the clean exploration model in Section 5.2

5.2 Exploration Model Cleaning

For those non-leaf behaviors, we have to clean their *main* functions by wrapping those statements into sub-behaviors and put the newly built sub-behaviors into *par, fsm, pipeline,* or *seq* statements. Since we do not explore the parallelism of the model yet, we model each behavior into *Finite State Machine (fsm)* fashion. We divided the statements into five categories and clean them in different manners:

- **common statements:** Common statements are those trivial, uncompound statements like variable assignments, function calls, etc. For these statements, we simply wrap them into a leaf sub-behavior with proper input/output ports.

- **if-branch:** If-branches usually contain one branch condition statement and one or two branch statements. We could recode the branches into different states and let the states change according to the condition statement. Figure 12 shows an example for *if-branch* cleaning.

- **for-loop:** For-loops usually contain an iteration variable and a for-loop iteration body. The for-loop will first initialize the iteration value, change the value per iteration and check the value to see when to terminate. We clean for-loops into several states by different sub-behaviors, one sub-behavior for iteration variable initialization, one sub-behavior for the iteration body, one sub-behavior for iteration variable changing and test the termination condition to determine the state transition. Figure 13 shows an example for *for-loop* cleaning.

- **while-loop:** While-loops are somewhat similar to the for-loops. The execution of the while-loop body is determined by the while-loop conditions. We clean the while-loop in the similar manner as for-loop cleaning. The while-loop body (while-loop condition changing is done in the loop body) is wrapped as a sub-behavior and a dummy sub-behavior is created for condition checking in *fsm* statement. Figure 14 shows an example for *while-loop* cleaning.

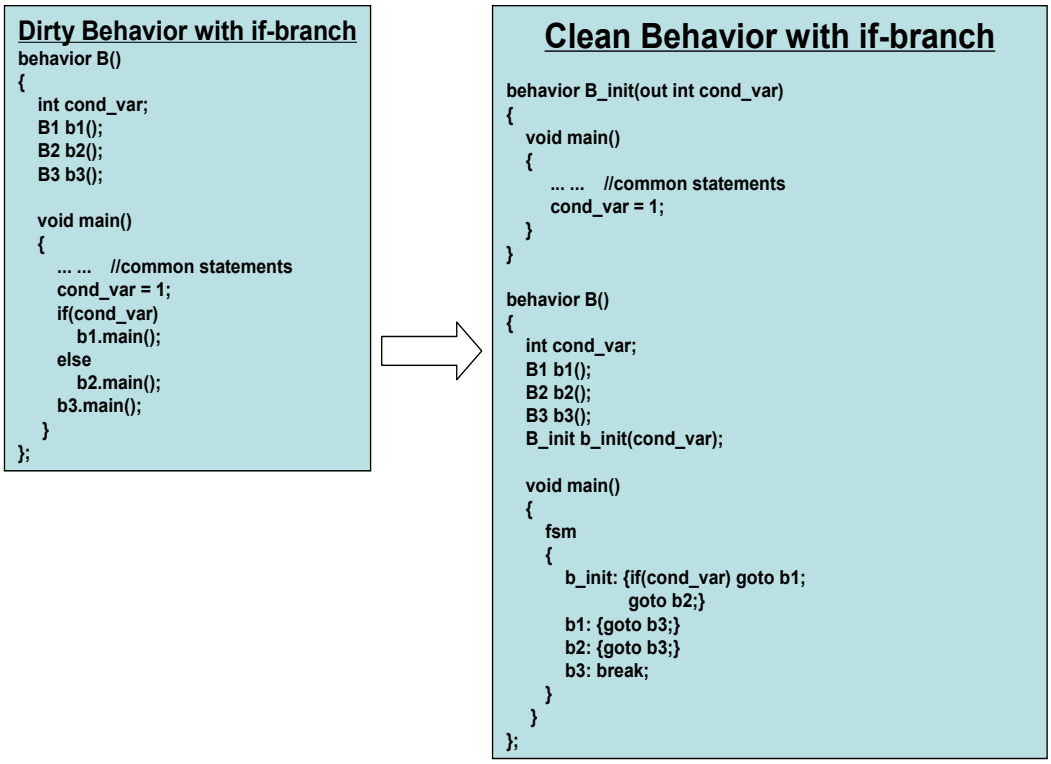


Figure 12: Example for if-branch cleaning

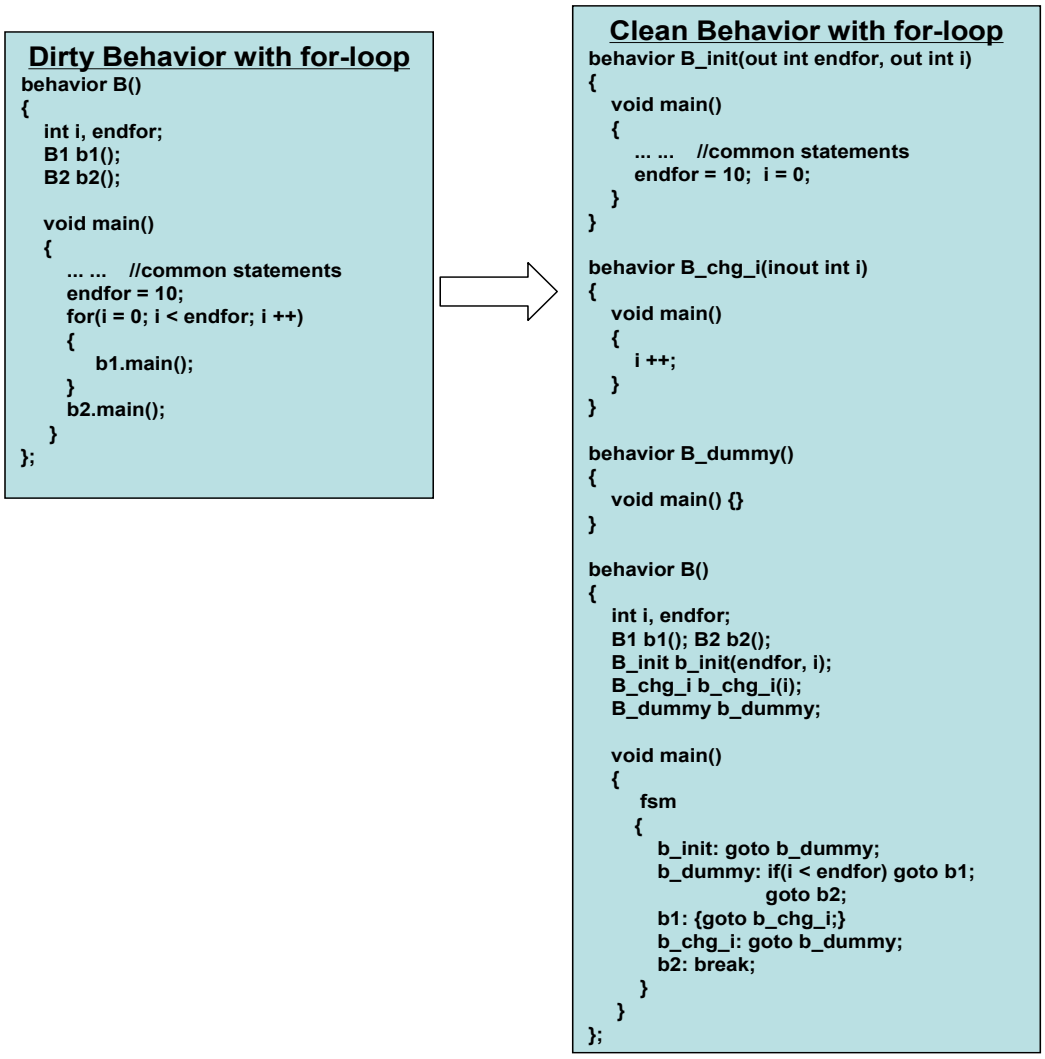


Figure 13: Example for for-loop cleaning

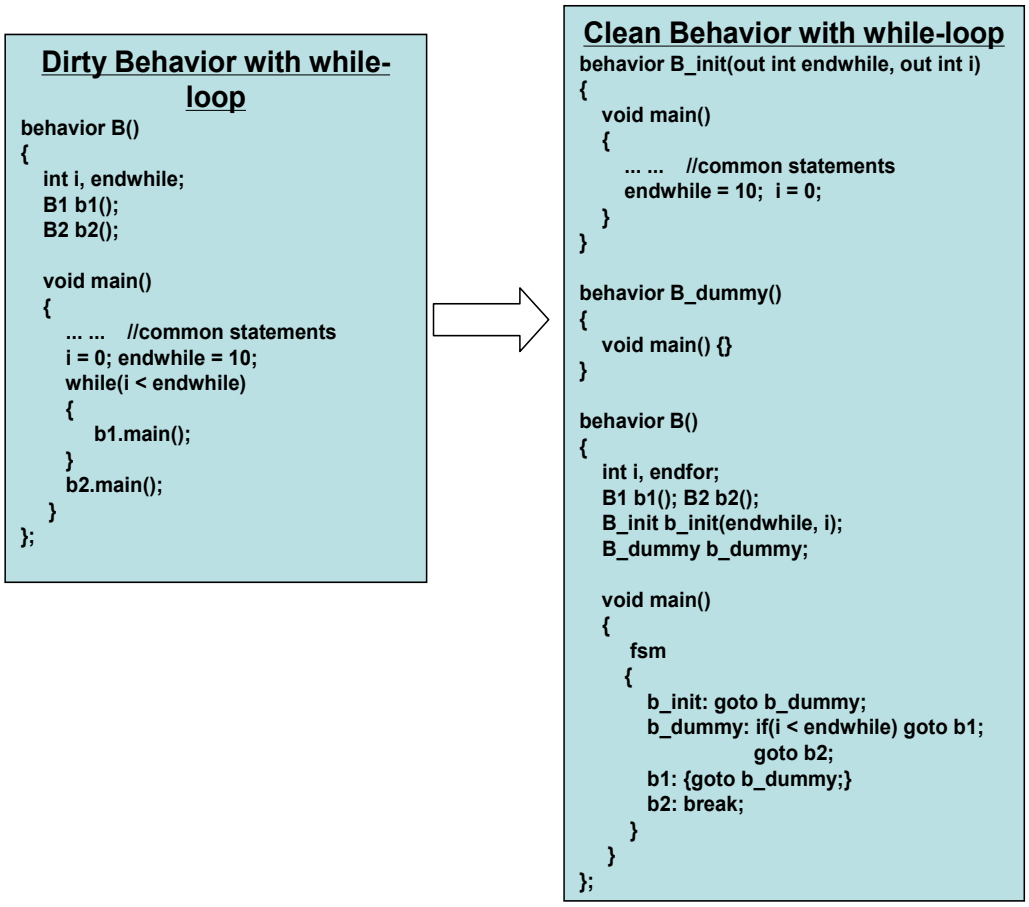


Figure 14: Example for while-loop cleaning

- **switch cases:** Switch cases are used for simplifying the if-branches under multiple different possible conditions. Switch cases could be recoded into *if-elseif-else* statements and then clean it as what is done for if-branches. Figure 15 shows an example for *switch cases* cleaning.

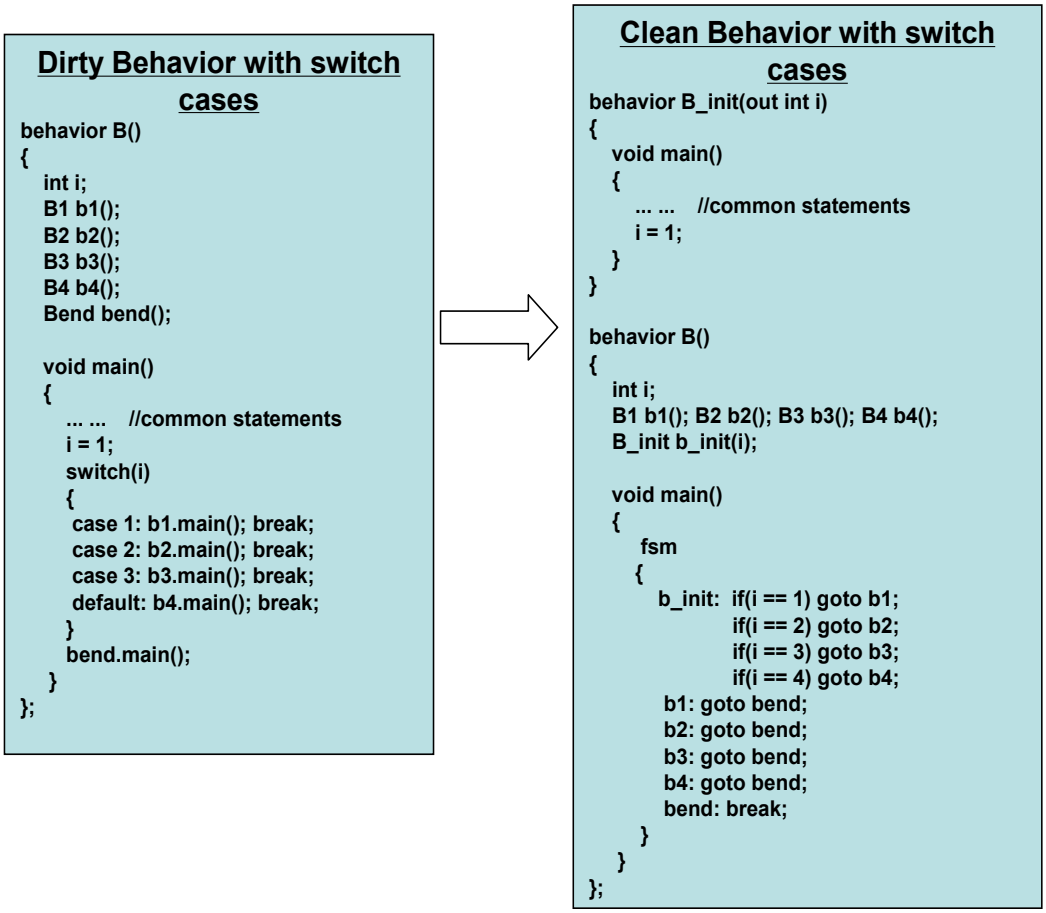


Figure 15: Example for switch cases cleaning

The body of the *if-branches*, *for-loops*, *while-loops* and *switch cases* may not be clean themselves, in other words they may contain both statements and behavior executions. We could apply our clean approach recursively to make them totally clean.

5.3 H.264 Decoder Exploration Model

We manually clean our H.264 decoder specification model which is discussed in Section 4 in bottom-up fashion. Figure 16 shows a non-leaf behavior tree before cleaning and after cleaning. A clean behavior hierarchy significantly is more complex than the original one.

Figure 16: A non-leaf behavior before and after cleaning

Our clean exploration specification model consists of 173 behaviors. Appendix A.5 shows the whole spanning behavior tree of our clean H.264 decoder model.

5.4 Simulation Result

We compiled our H.264 decoder clean exploration model on Linux kernel 2.6 by SpecC Compiler (SCC) V.2.2.1. The model could be compiled and runs well on Linux 2.6.

Appendix A.4 shows the simulation output of the H.264 specification model with our test file. It shows that the decoding execution is correct by those zero SNR values. The total number of frames in the test file is 299 and the system decoding time is 4.1 seconds on the host machine (PC with Intel(R) Pentium(R) 4 CPU 3.00GHz).

6 H.264 Decoder Initial Platform Model

In designing a video chip, one of the key questions is how to partition the hardware and software so as to maximize performance and minimize cost. In order to reduce the overall execution time, those behaviors which consume a high amount of time, have to be identified and mapped to high-speed hardware.

In this section, we will base on the specification model in Section 5, partition the behaviors onto Processing Elements (PE), and build an initial H.264 platform model. We use SCE tool to carry out above tasks.

6.1 Processing Elements Mapping

Current embedded multimedia applications are more and more implemented as processor based systems. For this initial platform model of H.264 decoder, we select a processor from the PE library provided by the SCE tool as the software component. At this instance, ARM_7TDMI with 100MHz maximum clock frequency is our choice and we map the whole DUT onto it. We simulate this pure software implementation with a 299 frame length H.264 video stream of 176×144 frame size and identify four critical time consuming behaviors, b_itrans4x4, b_itrans8x8, b_decode_one_macroblock, and b_intrapred_chroma shown in Figure 17. Total system time is estimated at 43.4s for pure software implementation, which is clearly beyond the 10 seconds real-time requirement.

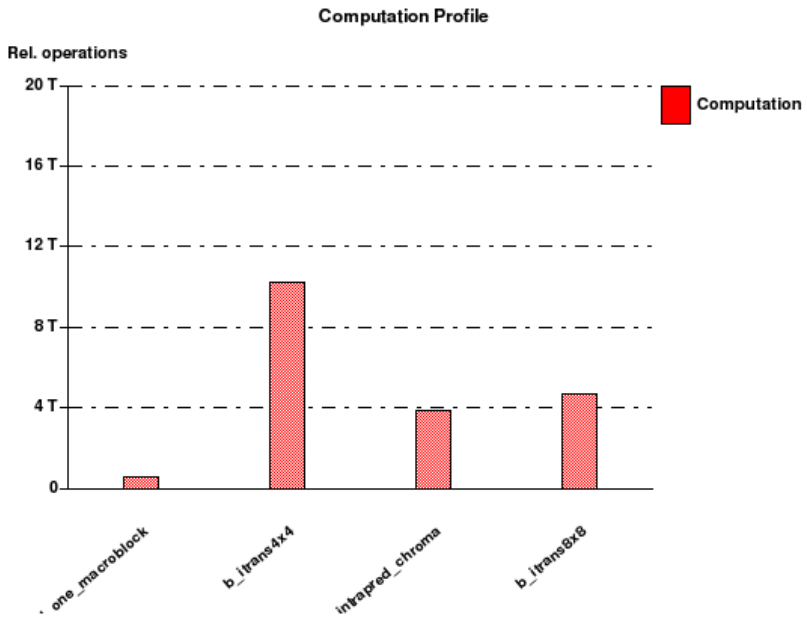


Figure 17: Computation profile of the time consuming behaviors

The timing and computation quantities of above four behaviors that take the largest part of total system time are shown in Figure 18, 19, 20 and 21.

Name	Type	N	Code	Computation	Data	Memory	Connections	Traffic
B_itrans4x4		429496	1100 B	13 s	256 B	56 B	3616 B	29.21 MB
bitdepth_luma_qp_scale	in int						4 B	0.00 MB
cof	in int [16][16]						1024 B	0.00 MB
ioff	in int						4 B	17.18 MB
joff	in int						4 B	6.87 MB
lossless_qpprime_flag	in int						4 B	0.00 MB
m7	inout int [16][16]						2048 B	0.00 MB
max_imgpel_value	in int						4 B	1.72 MB
max_imgpel_value_uv	in int						4 B	1.72 MB
mpr	in unsigned short int [16][16]						512 B	0.00 MB
qp	in int						4 B	0.00 MB
yuv	in int						4 B	1.72 MB
tmp	int [64]				256 B	0 B	0 B	0.00 MB

Figure 18: Timing and computation quantities of b_itrans4x4

Name	Type	N	Code	Computation	Data	Memory	Connections	Traffic
B_itrans8x8		55500	1776 B	6 s	256 B	104 B	10 B	33.85 MB
img	inout struct img_par *						2 B	29.53 MB
ioff	in int						4 B	3.55 MB
joff	in int						4 B	0.44 MB
pl	in ColorPlane						0 B	0.33 MB
tmp	int [64]				256 B	0 B	0 B	0.00 MB

Figure 19: Timing and computation quantities of b_itrans8x8

We proceed to speed our system up with custom hardware. The selected hardware element is HW_Standard from the SCE PE library and its clock frequency is set to 200MHz. We map b_itrans4x4, b_itrans8x8, b_decode_one_macroblock, and b_intrapred_chroma onto four copies of the hardware unit and summarize the PE allocation and software/hardware partition for this initial H.264 platform model in Table 2.

6.2 Architecture Refinement

Then we perform architecture refinement with SCE tool and evaluate above configuration. Figure 22 shows the behavior tree after architecture refinement and Figure 23 illustrates the hierarchical structure of the refined model. For the same H.264 file, we now obtain an estimated execution time of the software/hardware codesign model of 15.68 seconds. Taking into account the 299 frames in

Name	Type	N	Code	Computation	Data	Memory	Connections	Traffic
B_read_one_macroblock		29601	19004 B	4 s	0 B	66 B	4 B	0 B
p_currMB	in struct macroblock *						1 B	0 B
p_img	inout struct img_par *						2 B	0 B
p_inp	in struct inp_par *						1 B	0 B

Figure 20: Timing and computation quantities of b_decode_one_macroblock

Name	Type	N	Code	Computation	Data	Memory	Connections	Traffic
B_intrapred_chroma		40120	2740 B	1 s	0 B	798 B	8 B	4.55 MB
currMB	inout struct macroblock *						2 B	0.16 MB
img	inout struct img_par *						2 B	1.55 MB
uv	in int						4 B	2.83 MB

Figure 21: Timing and computation quantities of b_intrapred_chroma

	PE	Clock Frequency	Mapped Behavior
Software	ARM_7TDMI	100MHz	H264_DUT
Hardware	HW_Standard	200MHz	b_itrans4x4
Hardware	HW_Standard	200MHz	b_itrans8x8
Hardware	HW_Standard	200MHz	b_decode_one_macroblock
Hardware	HW_Standard	200MHz	b_intrapred_chroma

Table 2: Software/hardware configuration of initial platform model

the test system, our decoding speed results in 19.1fps, which is slower than the frame rate of 30fps required by National Television System Committee (NTSC) in United States.

However, there is still room to improve this initial platform model. We have the following choices to increase the decoding speed to meet the NSTC requirement:

- enhance the clock frequency of PEs (but this way the power consumption will be higher)
- find a better hardware and software partition so as to achieve better performance
- parallize some parts of the program which consume a large amount of time
- change the software component into a more powerful one

We will address these and other improvements in the near future.

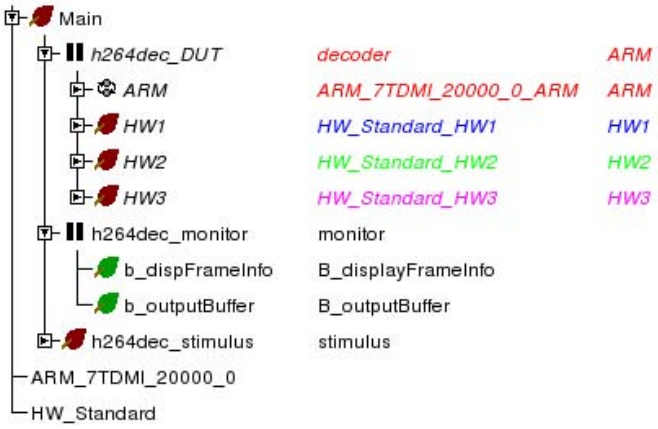


Figure 22: Behavior tree of H.264 decoder after architecture refinement

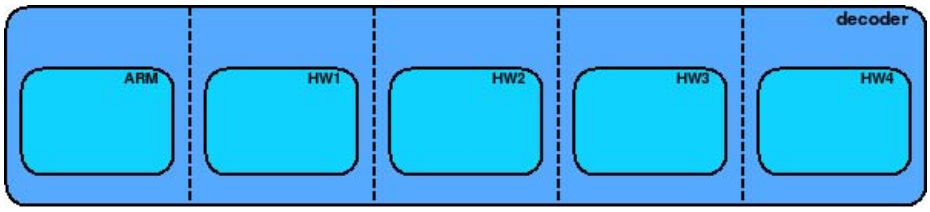


Figure 23: Hierarchical structure of initial platform model after architecture refinement

7 Conclusion and Future Work

In this project, we have successfully translated the H.264 decoder from C to SpecC. The original sequential algorithmic description of the system is replaced by a structured high level specification. Computation and communication information are isolated and encapsulated separately. The relationship between them can be refined during the later design space exploration. Based on different target systems, different topology of these encapsulated computation elements can be tried. The optimal one will be chosen as the target architecture, and later on can be synthesized into lower level description language, such as VHDL or Verilog.

For future work, we plan to synthesize the high level specification onto a FPGA board to test the integrity of the process. Based on the real data obtained from the hardware, adjustments can be made to refine the high level model. Power characteristics can also be added to the high level specification. With this information, the trade off decision of the system architecture can be made earlier and more accurate at the higher level.

References

[1] L. Cai, A. Gerstlauer, S. Abdi, J. Peng, D. Shin, H. Yu, R. Dömer, and D. Gajski. System-on-chip environment (SCE version 2.2.0 beta): Manual. Technical Report CECS-TR-03-45, Center for Embedded Computer Systems, University of California, Irvine, December 2003.

[2] P. Chandraiah and R. Dömer. Specification and design of a MP3 audio decoder. Technical Report CECS-TR-05-04, Center for Embedded Computer Systems, University of California, Irvine, May 2005.

[3] P. Chandraiah, H. Schirner, N. Srinivas, and R. Dömer. System-on chip modeling and design: A case study on MP3 decoder. Technical Report CECS-TR-04-17, Center for Embedded Computer Systems, University of California, Irvine, June 2004.

[4] Rainer Dömer, Andreas Gerstlauer, and Daniel Gajski. *SpecC Language Reference Manual, Version 2.0*. SpecC Technology Open Consortium, http://www.specc.org, December 2002.

[5] Andreas Gerstlauer, Rainer Dömer, Junyu Peng, and Daniel D. Gajski. *System Design: A Practical Guide with SpecC*. Kluwer Academic Publishers, 2001.

[6] HHI. H.264/AVC JM Reference Software. . http://iphome.hhi.de/suehring/tml/.

[7] Joint Video Team of ITU-T and ISO/IEC JTC 1. *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 — ISO/IEC 14496-10 AVC)*. Document JVT-G050r1, 2003.

[8] Iain E G Richardson. H.264/MPEG-4 Part 10 White Paper. http://www.vcodex.com/, 2002.

[9] Wikipedia H.264 Std. . http://en.wikipedia.org/wiki/H264.

A Appendix

A.1 Decoder configuration file *decoder.cfg*

```
test.264                ........H.264/AVC coded bitstream
test_dec.yuv            ........Output file, YUV/RGB
test_rec.yuv            ........Ref sequence (for SNR)
1                       ........Write 4:2:0 chroma components for monochrome streams
1                       ........NAL mode (0=Annex B, 1: RTP packets)
0                       ........SNR computation offset
2                       ........Poc Scale (1 or 2)
500000                  ........Rate_Decoder
104000                  ........B_decoder
73000                   ........F_decoder
leakybucketparam.cfg    ........LeakyBucket Params
2                       ........Err Concealment(0:Off,1:Frame Copy,2:Motion Copy)
2                       ........Reference POC gap (2: IPP (Default), 4: IbP / IpP)
2                       ........POC gap (2: IPP /IbP/IpP (Default), 4: IPP with frame
    skip = 1 etc.)
0                       ........Silent decode

This is a file containing input parameters to the JVT H.264/AVC decoder.
The text line following each parameter is discarded by the decoder.

For bug reporting and known issues see:
https://ipbt.hhi.de
```

A.2 Simulation result of the JM C reference code

```
------------------------------------ JM 13.0 (FREXt) ------------------------------------
 Decoder config file                          : decoder.cfg
-----------------------------------------------------------------------------------------
 Input H.264 bitstream                        : test.264
 Output decoded YUV                           : test_dec.yuv
 Output status file                           : log.dec
 Input reference file                         : test_rec.yuv
-----------------------------------------------------------------------------------------
POC must = frame# or field# for SNRs to be correct
-----------------------------------------------------------------------------------------
  Frame          POC   Pic#   QP     SnrY     SnrU     SnrV    Y:U:V  Time(ms)
-----------------------------------------------------------------------------------------
00000(IDR)         0      0   28   0.0000   0.0000   0.0000   4:2:0        10
00002( P )         4      1   28   0.0000   0.0000   0.0000   4:2:0        11
00001( b )         2      2   30   0.0000   0.0000   0.0000   4:2:0        12
00004( P )         8      2   28   0.0000   0.0000   0.0000   4:2:0         9
00003( b )         6      3   30   0.0000   0.0000   0.0000   4:2:0         9
00006( P )        12      3   28   0.0000   0.0000   0.0000   4:2:0        11
00005( b )        10      4   30   0.0000   0.0000   0.0000   4:2:0        13
00008( P )        16      4   28   0.0000   0.0000   0.0000   4:2:0        10
00007( b )        14      5   30   0.0000   0.0000   0.0000   4:2:0         9
00010( P )        20      5   28   0.0000   0.0000   0.0000   4:2:0        11
00009( b )        18      6   30   0.0000   0.0000   0.0000   4:2:0        13
00012( P )        24      6   28   0.0000   0.0000   0.0000   4:2:0         9
00011( b )        22      7   30   0.0000   0.0000   0.0000   4:2:0         9
00014( P )        28      7   28   0.0000   0.0000   0.0000   4:2:0        11
00013( b )        26      8   30   0.0000   0.0000   0.0000   4:2:0        13
00016( P )        32      8   28   0.0000   0.0000   0.0000   4:2:0        10
00015( b )        30      9   30   0.0000   0.0000   0.0000   4:2:0         9
00018( P )        36      9   28   0.0000   0.0000   0.0000   4:2:0        12
```

00017(b)	34	10	30	0.0000	0.0000	0.0000	4:2:0	13
00020(P)	40	10	28	0.0000	0.0000	0.0000	4:2:0	10
00019(b)	38	11	30	0.0000	0.0000	0.0000	4:2:0	10
00022(P)	44	11	28	0.0000	0.0000	0.0000	4:2:0	11
00021(b)	42	12	30	0.0000	0.0000	0.0000	4:2:0	13
00024(P)	48	12	28	0.0000	0.0000	0.0000	4:2:0	10
00023(b)	46	13	30	0.0000	0.0000	0.0000	4:2:0	9
00026(P)	52	13	28	0.0000	0.0000	0.0000	4:2:0	11
00025(b)	50	14	30	0.0000	0.0000	0.0000	4:2:0	13
00028(P)	56	14	28	0.0000	0.0000	0.0000	4:2:0	12
00027(b)	54	15	30	0.0000	0.0000	0.0000	4:2:0	9
00030(P)	60	15	28	0.0000	0.0000	0.0000	4:2:0	12
00029(b)	58	0	30	0.0000	0.0000	0.0000	4:2:0	13
00032(P)	64	0	28	0.0000	0.0000	0.0000	4:2:0	12
00031(b)	62	1	30	0.0000	0.0000	0.0000	4:2:0	9
00034(P)	68	1	28	0.0000	0.0000	0.0000	4:2:0	11
00033(b)	66	2	30	0.0000	0.0000	0.0000	4:2:0	13
00036(P)	72	2	28	0.0000	0.0000	0.0000	4:2:0	12
00035(b)	70	3	30	0.0000	0.0000	0.0000	4:2:0	9
00038(P)	76	3	28	0.0000	0.0000	0.0000	4:2:0	12
00037(b)	74	4	30	0.0000	0.0000	0.0000	4:2:0	13
00040(P)	80	4	28	0.0000	0.0000	0.0000	4:2:0	10
00039(b)	78	5	30	0.0000	0.0000	0.0000	4:2:0	10
00042(P)	84	5	28	0.0000	0.0000	0.0000	4:2:0	11
00041(b)	82	6	30	0.0000	0.0000	0.0000	4:2:0	12
00044(P)	88	6	28	0.0000	0.0000	0.0000	4:2:0	11
00043(b)	86	7	30	0.0000	0.0000	0.0000	4:2:0	9
00046(P)	92	7	28	0.0000	0.0000	0.0000	4:2:0	11
00045(b)	90	8	30	0.0000	0.0000	0.0000	4:2:0	13
00048(P)	96	8	28	0.0000	0.0000	0.0000	4:2:0	13
00047(b)	94	9	30	0.0000	0.0000	0.0000	4:2:0	9
00050(P)	100	9	28	0.0000	0.0000	0.0000	4:2:0	11
00049(b)	98	10	30	0.0000	0.0000	0.0000	4:2:0	14
00052(P)	104	10	28	0.0000	0.0000	0.0000	4:2:0	11
00051(b)	102	11	30	0.0000	0.0000	0.0000	4:2:0	9
00054(P)	108	11	28	0.0000	0.0000	0.0000	4:2:0	12
00053(b)	106	12	30	0.0000	0.0000	0.0000	4:2:0	13
00056(P)	112	12	28	0.0000	0.0000	0.0000	4:2:0	12
00055(b)	110	13	30	0.0000	0.0000	0.0000	4:2:0	10
00058(P)	116	13	28	0.0000	0.0000	0.0000	4:2:0	11
00057(b)	114	14	30	0.0000	0.0000	0.0000	4:2:0	14
00060(P)	120	14	28	0.0000	0.0000	0.0000	4:2:0	12
00059(b)	118	15	30	0.0000	0.0000	0.0000	4:2:0	10
00062(P)	124	15	28	0.0000	0.0000	0.0000	4:2:0	11
00061(b)	122	0	30	0.0000	0.0000	0.0000	4:2:0	14
00064(P)	128	0	28	0.0000	0.0000	0.0000	4:2:0	12
00063(b)	126	1	30	0.0000	0.0000	0.0000	4:2:0	10
00066(P)	132	1	28	0.0000	0.0000	0.0000	4:2:0	12
00065(b)	130	2	30	0.0000	0.0000	0.0000	4:2:0	13
00068(P)	136	2	28	0.0000	0.0000	0.0000	4:2:0	12
00067(b)	134	3	30	0.0000	0.0000	0.0000	4:2:0	9
00070(P)	140	3	28	0.0000	0.0000	0.0000	4:2:0	11
00069(b)	138	4	30	0.0000	0.0000	0.0000	4:2:0	14
00072(P)	144	4	28	0.0000	0.0000	0.0000	4:2:0	12
00071(b)	142	5	30	0.0000	0.0000	0.0000	4:2:0	9
00074(P)	148	5	28	0.0000	0.0000	0.0000	4:2:0	11
00073(b)	146	6	30	0.0000	0.0000	0.0000	4:2:0	14
00076(P)	152	6	28	0.0000	0.0000	0.0000	4:2:0	13
00075(b)	150	7	30	0.0000	0.0000	0.0000	4:2:0	9
00078(P)	156	7	28	0.0000	0.0000	0.0000	4:2:0	11

00077(b)	154	8	30	0.0000	0.0000	0.0000	4:2:0	13
00080(P)	160	8	28	0.0000	0.0000	0.0000	4:2:0	14
00079(b)	158	9	30	0.0000	0.0000	0.0000	4:2:0	9
00082(P)	164	9	28	0.0000	0.0000	0.0000	4:2:0	11
00081(b)	162	10	30	0.0000	0.0000	0.0000	4:2:0	14
00084(P)	168	10	28	0.0000	0.0000	0.0000	4:2:0	12
00083(b)	166	11	30	0.0000	0.0000	0.0000	4:2:0	9
00086(P)	172	11	28	0.0000	0.0000	0.0000	4:2:0	11
00085(b)	170	12	30	0.0000	0.0000	0.0000	4:2:0	13
00088(P)	176	12	28	0.0000	0.0000	0.0000	4:2:0	13
00087(b)	174	13	30	0.0000	0.0000	0.0000	4:2:0	10
00090(P)	180	13	28	0.0000	0.0000	0.0000	4:2:0	11
00089(b)	178	14	30	0.0000	0.0000	0.0000	4:2:0	14
00092(P)	184	14	28	0.0000	0.0000	0.0000	4:2:0	11
00091(b)	182	15	30	0.0000	0.0000	0.0000	4:2:0	10
00094(P)	188	15	28	0.0000	0.0000	0.0000	4:2:0	11
00093(b)	186	0	30	0.0000	0.0000	0.0000	4:2:0	13
00096(P)	192	0	28	0.0000	0.0000	0.0000	4:2:0	13
00095(b)	190	1	30	0.0000	0.0000	0.0000	4:2:0	9
00098(P)	196	1	28	0.0000	0.0000	0.0000	4:2:0	11
00097(b)	194	2	30	0.0000	0.0000	0.0000	4:2:0	14
00100(P)	200	2	28	0.0000	0.0000	0.0000	4:2:0	12
00099(b)	198	3	30	0.0000	0.0000	0.0000	4:2:0	9
00102(P)	204	3	28	0.0000	0.0000	0.0000	4:2:0	12
00101(b)	202	4	30	0.0000	0.0000	0.0000	4:2:0	13
00104(P)	208	4	28	0.0000	0.0000	0.0000	4:2:0	11
00103(b)	206	5	30	0.0000	0.0000	0.0000	4:2:0	9
00106(P)	212	5	28	0.0000	0.0000	0.0000	4:2:0	11
00105(b)	210	6	30	0.0000	0.0000	0.0000	4:2:0	14
00108(P)	216	6	28	0.0000	0.0000	0.0000	4:2:0	13
00107(b)	214	7	30	0.0000	0.0000	0.0000	4:2:0	10
00110(P)	220	7	28	0.0000	0.0000	0.0000	4:2:0	11
00109(b)	218	8	30	0.0000	0.0000	0.0000	4:2:0	13
00112(P)	224	8	28	0.0000	0.0000	0.0000	4:2:0	11
00111(b)	222	9	30	0.0000	0.0000	0.0000	4:2:0	9
00114(P)	228	9	28	0.0000	0.0000	0.0000	4:2:0	11
00113(b)	226	10	30	0.0000	0.0000	0.0000	4:2:0	13
00116(P)	232	10	28	0.0000	0.0000	0.0000	4:2:0	12
00115(b)	230	11	30	0.0000	0.0000	0.0000	4:2:0	10
00118(P)	236	11	28	0.0000	0.0000	0.0000	4:2:0	11
00117(b)	234	12	30	0.0000	0.0000	0.0000	4:2:0	13
00120(P)	240	12	28	0.0000	0.0000	0.0000	4:2:0	13
00119(b)	238	13	30	0.0000	0.0000	0.0000	4:2:0	10
00122(P)	244	13	28	0.0000	0.0000	0.0000	4:2:0	11
00121(b)	242	14	30	0.0000	0.0000	0.0000	4:2:0	14
00124(P)	248	14	28	0.0000	0.0000	0.0000	4:2:0	13
00123(b)	246	15	30	0.0000	0.0000	0.0000	4:2:0	10
00126(P)	252	15	28	0.0000	0.0000	0.0000	4:2:0	11
00125(b)	250	0	30	0.0000	0.0000	0.0000	4:2:0	14
00128(P)	256	0	28	0.0000	0.0000	0.0000	4:2:0	13
00127(b)	254	1	30	0.0000	0.0000	0.0000	4:2:0	9
00130(P)	260	1	28	0.0000	0.0000	0.0000	4:2:0	12
00129(b)	258	2	30	0.0000	0.0000	0.0000	4:2:0	14
00132(P)	264	2	28	0.0000	0.0000	0.0000	4:2:0	13
00131(b)	262	3	30	0.0000	0.0000	0.0000	4:2:0	10
00134(P)	268	3	28	0.0000	0.0000	0.0000	4:2:0	11
00133(b)	266	4	30	0.0000	0.0000	0.0000	4:2:0	14
00136(P)	272	4	28	0.0000	0.0000	0.0000	4:2:0	12
00135(b)	270	5	30	0.0000	0.0000	0.0000	4:2:0	10
00138(P)	276	5	28	0.0000	0.0000	0.0000	4:2:0	11

00137(b)	274	6	30	0.0000	0.0000	0.0000	4:2:0	14
00140(P)	280	6	28	0.0000	0.0000	0.0000	4:2:0	13
00139(b)	278	7	30	0.0000	0.0000	0.0000	4:2:0	10
00142(P)	284	7	28	0.0000	0.0000	0.0000	4:2:0	11
00141(b)	282	8	30	0.0000	0.0000	0.0000	4:2:0	14
00144(P)	288	8	28	0.0000	0.0000	0.0000	4:2:0	13
00143(b)	286	9	30	0.0000	0.0000	0.0000	4:2:0	9
00146(P)	292	9	28	0.0000	0.0000	0.0000	4:2:0	12
00145(b)	290	10	30	0.0000	0.0000	0.0000	4:2:0	13
00148(P)	296	10	28	0.0000	0.0000	0.0000	4:2:0	13
00147(b)	294	11	30	0.0000	0.0000	0.0000	4:2:0	10
00150(P)	300	11	28	0.0000	0.0000	0.0000	4:2:0	12
00149(b)	298	12	30	0.0000	0.0000	0.0000	4:2:0	14
00152(P)	304	12	28	0.0000	0.0000	0.0000	4:2:0	13
00151(b)	302	13	30	0.0000	0.0000	0.0000	4:2:0	9
00154(P)	308	13	28	0.0000	0.0000	0.0000	4:2:0	12
00153(b)	306	14	30	0.0000	0.0000	0.0000	4:2:0	14
00156(P)	312	14	28	0.0000	0.0000	0.0000	4:2:0	13
00155(b)	310	15	30	0.0000	0.0000	0.0000	4:2:0	10
00158(P)	316	15	28	0.0000	0.0000	0.0000	4:2:0	12
00157(b)	314	0	30	0.0000	0.0000	0.0000	4:2:0	14
00160(P)	320	0	28	0.0000	0.0000	0.0000	4:2:0	13
00159(b)	318	1	30	0.0000	0.0000	0.0000	4:2:0	10
00162(P)	324	1	28	0.0000	0.0000	0.0000	4:2:0	11
00161(b)	322	2	30	0.0000	0.0000	0.0000	4:2:0	14
00164(P)	328	2	28	0.0000	0.0000	0.0000	4:2:0	13
00163(b)	326	3	30	0.0000	0.0000	0.0000	4:2:0	10
00166(P)	332	3	28	0.0000	0.0000	0.0000	4:2:0	12
00165(b)	330	4	30	0.0000	0.0000	0.0000	4:2:0	13
00168(P)	336	4	28	0.0000	0.0000	0.0000	4:2:0	11
00167(b)	334	5	30	0.0000	0.0000	0.0000	4:2:0	10
00170(P)	340	5	28	0.0000	0.0000	0.0000	4:2:0	11
00169(b)	338	6	30	0.0000	0.0000	0.0000	4:2:0	15
00172(P)	344	6	28	0.0000	0.0000	0.0000	4:2:0	13
00171(b)	342	7	30	0.0000	0.0000	0.0000	4:2:0	10
00174(P)	348	7	28	0.0000	0.0000	0.0000	4:2:0	12
00173(b)	346	8	30	0.0000	0.0000	0.0000	4:2:0	14
00176(P)	352	8	28	0.0000	0.0000	0.0000	4:2:0	14
00175(b)	350	9	30	0.0000	0.0000	0.0000	4:2:0	10
00178(P)	356	9	28	0.0000	0.0000	0.0000	4:2:0	12
00177(b)	354	10	30	0.0000	0.0000	0.0000	4:2:0	15
00180(P)	360	10	28	0.0000	0.0000	0.0000	4:2:0	14
00179(b)	358	11	30	0.0000	0.0000	0.0000	4:2:0	9
00182(P)	364	11	28	0.0000	0.0000	0.0000	4:2:0	12
00181(b)	362	12	30	0.0000	0.0000	0.0000	4:2:0	16
00184(P)	368	12	28	0.0000	0.0000	0.0000	4:2:0	14
00183(b)	366	13	30	0.0000	0.0000	0.0000	4:2:0	9
00186(P)	372	13	28	0.0000	0.0000	0.0000	4:2:0	13
00185(b)	370	14	30	0.0000	0.0000	0.0000	4:2:0	14
00188(P)	376	14	28	0.0000	0.0000	0.0000	4:2:0	15
00187(b)	374	15	30	0.0000	0.0000	0.0000	4:2:0	9
00190(P)	380	15	28	0.0000	0.0000	0.0000	4:2:0	13
00189(b)	378	0	30	0.0000	0.0000	0.0000	4:2:0	15
00192(P)	384	0	28	0.0000	0.0000	0.0000	4:2:0	15
00191(b)	382	1	30	0.0000	0.0000	0.0000	4:2:0	9
00194(P)	388	1	28	0.0000	0.0000	0.0000	4:2:0	12
00193(b)	386	2	30	0.0000	0.0000	0.0000	4:2:0	15
00196(P)	392	2	28	0.0000	0.0000	0.0000	4:2:0	13
00195(b)	390	3	30	0.0000	0.0000	0.0000	4:2:0	9
00198(P)	396	3	28	0.0000	0.0000	0.0000	4:2:0	13

00197(b)	394	4	30	0.0000	0.0000	0.0000	4:2:0	15
00200(P)	400	4	28	0.0000	0.0000	0.0000	4:2:0	15
00199(b)	398	5	30	0.0000	0.0000	0.0000	4:2:0	9
00202(P)	404	5	28	0.0000	0.0000	0.0000	4:2:0	13
00201(b)	402	6	30	0.0000	0.0000	0.0000	4:2:0	15
00204(P)	408	6	28	0.0000	0.0000	0.0000	4:2:0	15
00203(b)	406	7	30	0.0000	0.0000	0.0000	4:2:0	9
00206(P)	412	7	28	0.0000	0.0000	0.0000	4:2:0	12
00205(b)	410	8	30	0.0000	0.0000	0.0000	4:2:0	14
00208(P)	416	8	28	0.0000	0.0000	0.0000	4:2:0	14
00207(b)	414	9	30	0.0000	0.0000	0.0000	4:2:0	9
00210(P)	420	9	28	0.0000	0.0000	0.0000	4:2:0	13
00209(b)	418	10	30	0.0000	0.0000	0.0000	4:2:0	15
00212(P)	424	10	28	0.0000	0.0000	0.0000	4:2:0	14
00211(b)	422	11	30	0.0000	0.0000	0.0000	4:2:0	9
00214(P)	428	11	28	0.0000	0.0000	0.0000	4:2:0	12
00213(b)	426	12	30	0.0000	0.0000	0.0000	4:2:0	15
00216(P)	432	12	28	0.0000	0.0000	0.0000	4:2:0	12
00215(b)	430	13	30	0.0000	0.0000	0.0000	4:2:0	9
00218(P)	436	13	28	0.0000	0.0000	0.0000	4:2:0	12
00217(b)	434	14	30	0.0000	0.0000	0.0000	4:2:0	14
00220(P)	440	14	28	0.0000	0.0000	0.0000	4:2:0	14
00219(b)	438	15	30	0.0000	0.0000	0.0000	4:2:0	9
00222(P)	444	15	28	0.0000	0.0000	0.0000	4:2:0	11
00221(b)	442	0	30	0.0000	0.0000	0.0000	4:2:0	16
00224(P)	448	0	28	0.0000	0.0000	0.0000	4:2:0	13
00223(b)	446	1	30	0.0000	0.0000	0.0000	4:2:0	9
00226(P)	452	1	28	0.0000	0.0000	0.0000	4:2:0	11
00225(b)	450	2	30	0.0000	0.0000	0.0000	4:2:0	15
00228(P)	456	2	28	0.0000	0.0000	0.0000	4:2:0	15
00227(b)	454	3	30	0.0000	0.0000	0.0000	4:2:0	9
00230(P)	460	3	28	0.0000	0.0000	0.0000	4:2:0	12
00229(b)	458	4	30	0.0000	0.0000	0.0000	4:2:0	14
00232(P)	464	4	28	0.0000	0.0000	0.0000	4:2:0	15
00231(b)	462	5	30	0.0000	0.0000	0.0000	4:2:0	10
00234(P)	468	5	28	0.0000	0.0000	0.0000	4:2:0	12
00233(b)	466	6	30	0.0000	0.0000	0.0000	4:2:0	15
00236(P)	472	6	28	0.0000	0.0000	0.0000	4:2:0	14
00235(b)	470	7	30	0.0000	0.0000	0.0000	4:2:0	9
00238(P)	476	7	28	0.0000	0.0000	0.0000	4:2:0	13
00237(b)	474	8	30	0.0000	0.0000	0.0000	4:2:0	15
00240(P)	480	8	28	0.0000	0.0000	0.0000	4:2:0	14
00239(b)	478	9	30	0.0000	0.0000	0.0000	4:2:0	9
00242(P)	484	9	28	0.0000	0.0000	0.0000	4:2:0	12
00241(b)	482	10	30	0.0000	0.0000	0.0000	4:2:0	14
00244(P)	488	10	28	0.0000	0.0000	0.0000	4:2:0	16
00243(b)	486	11	30	0.0000	0.0000	0.0000	4:2:0	10
00246(P)	492	11	28	0.0000	0.0000	0.0000	4:2:0	11
00245(b)	490	12	30	0.0000	0.0000	0.0000	4:2:0	15
00248(P)	496	12	28	0.0000	0.0000	0.0000	4:2:0	13
00247(b)	494	13	30	0.0000	0.0000	0.0000	4:2:0	9
00250(P)	500	13	28	0.0000	0.0000	0.0000	4:2:0	12
00249(b)	498	14	30	0.0000	0.0000	0.0000	4:2:0	15
00252(P)	504	14	28	0.0000	0.0000	0.0000	4:2:0	14
00251(b)	502	15	30	0.0000	0.0000	0.0000	4:2:0	10
00254(P)	508	15	28	0.0000	0.0000	0.0000	4:2:0	12
00253(b)	506	0	30	0.0000	0.0000	0.0000	4:2:0	14
00256(P)	512	0	28	0.0000	0.0000	0.0000	4:2:0	14
00255(b)	510	1	30	0.0000	0.0000	0.0000	4:2:0	12
00258(P)	516	1	28	0.0000	0.0000	0.0000	4:2:0	12

```
00257( b )      514     2    30   0.0000   0.0000   0.0000  4:2:0     14
00260( P )      520     2    28   0.0000   0.0000   0.0000  4:2:0     14
00259( b )      518     3    30   0.0000   0.0000   0.0000  4:2:0      9
00262( P )      524     3    28   0.0000   0.0000   0.0000  4:2:0     12
00261( b )      522     4    30   0.0000   0.0000   0.0000  4:2:0     14
00264( P )      528     4    28   0.0000   0.0000   0.0000  4:2:0     14
00263( b )      526     5    30   0.0000   0.0000   0.0000  4:2:0     10
00266( P )      532     5    28   0.0000   0.0000   0.0000  4:2:0     13
00265( b )      530     6    30   0.0000   0.0000   0.0000  4:2:0     15
00268( P )      536     6    28   0.0000   0.0000   0.0000  4:2:0     13
00267( b )      534     7    30   0.0000   0.0000   0.0000  4:2:0     11
00270( P )      540     7    28   0.0000   0.0000   0.0000  4:2:0     12
00269( b )      538     8    30   0.0000   0.0000   0.0000  4:2:0     14
00272( P )      544     8    28   0.0000   0.0000   0.0000  4:2:0     14
00271( b )      542     9    30   0.0000   0.0000   0.0000  4:2:0      9
00274( P )      548     9    28   0.0000   0.0000   0.0000  4:2:0     11
00273( b )      546    10    30   0.0000   0.0000   0.0000  4:2:0     14
00276( P )      552    10    28   0.0000   0.0000   0.0000  4:2:0     14
00275( b )      550    11    30   0.0000   0.0000   0.0000  4:2:0      9
00278( P )      556    11    28   0.0000   0.0000   0.0000  4:2:0     12
00277( b )      554    12    30   0.0000   0.0000   0.0000  4:2:0     14
00280( P )      560    12    28   0.0000   0.0000   0.0000  4:2:0     12
00279( b )      558    13    30   0.0000   0.0000   0.0000  4:2:0     10
00282( P )      564    13    28   0.0000   0.0000   0.0000  4:2:0     11
00281( b )      562    14    30   0.0000   0.0000   0.0000  4:2:0     14
00284( P )      568    14    28   0.0000   0.0000   0.0000  4:2:0     12
00283( b )      566    15    30   0.0000   0.0000   0.0000  4:2:0     10
00286( P )      572    15    28   0.0000   0.0000   0.0000  4:2:0     11
00285( b )      570     0    30   0.0000   0.0000   0.0000  4:2:0     14
00288( P )      576     0    28   0.0000   0.0000   0.0000  4:2:0     12
00287( b )      574     1    30   0.0000   0.0000   0.0000  4:2:0      9
00290( P )      580     1    28   0.0000   0.0000   0.0000  4:2:0     11
00289( b )      578     2    30   0.0000   0.0000   0.0000  4:2:0     14
00292( P )      584     2    28   0.0000   0.0000   0.0000  4:2:0     12
00291( b )      582     3    30   0.0000   0.0000   0.0000  4:2:0     10
00294( P )      588     3    28   0.0000   0.0000   0.0000  4:2:0     11
00293( b )      586     4    30   0.0000   0.0000   0.0000  4:2:0     13
00296( P )      592     4    28   0.0000   0.0000   0.0000  4:2:0     13
00295( b )      590     5    30   0.0000   0.0000   0.0000  4:2:0      9
00298( P )      596     5    28   0.0000   0.0000   0.0000  4:2:0     11
00297( b )      594     6    30   0.0000   0.0000   0.0000  4:2:0     13
------------------- Average SNR all frames -----------------------------
 SNR Y(dB)             :  0.00
 SNR U(dB)             :  0.00
 SNR V(dB)             :  0.00
 Total decoding time : 3.557 sec (84.060 fps)
-----------------------------------------------------------------------
 Exit JM 13 (FREXt) decoder, ver 13.0
```

A.3 Simulation result of the SpecC specification model

```
------------------------- JM 13.0 (FREXt) -----------------------------
 Decoder config file                    : decoder.cfg
-----------------------------------------------------------------------
 Input H.264 bitstream                  : test.264
 Output decoded YUV                     : test_dec.yuv
 Output status file                     : log.dec
 Input reference file                   : test_rec.yuv
-----------------------------------------------------------------------
POC must = frame# or field# for SNRs to be correct
```

Frame	POC	Pic#	*QP*	*SnrY*	*SnrU*	*SnrV*	*Y:U:V*	*Time(ms)*
00000(IDR)	0	0	28	0.0000	0.0000	0.0000	4:2:0	14
00002(P)	4	1	28	0.0000	0.0000	0.0000	4:2:0	15
00001(b)	2	2	30	0.0000	0.0000	0.0000	4:2:0	17
00004(P)	8	2	28	0.0000	0.0000	0.0000	4:2:0	12
00003(b)	6	3	30	0.0000	0.0000	0.0000	4:2:0	12
00006(P)	12	3	28	0.0000	0.0000	0.0000	4:2:0	15
00005(b)	10	4	30	0.0000	0.0000	0.0000	4:2:0	19
00008(P)	16	4	28	0.0000	0.0000	0.0000	4:2:0	13
00007(b)	14	5	30	0.0000	0.0000	0.0000	4:2:0	14
00010(P)	20	5	28	0.0000	0.0000	0.0000	4:2:0	14
00009(b)	18	6	30	0.0000	0.0000	0.0000	4:2:0	18
00012(P)	24	6	28	0.0000	0.0000	0.0000	4:2:0	12
00011(b)	22	7	30	0.0000	0.0000	0.0000	4:2:0	13
00014(P)	28	7	28	0.0000	0.0000	0.0000	4:2:0	14
00013(b)	26	8	30	0.0000	0.0000	0.0000	4:2:0	19
00016(P)	32	8	28	0.0000	0.0000	0.0000	4:2:0	13
00015(b)	30	9	30	0.0000	0.0000	0.0000	4:2:0	13
00018(P)	36	9	28	0.0000	0.0000	0.0000	4:2:0	16
00017(b)	34	10	30	0.0000	0.0000	0.0000	4:2:0	18
00020(P)	40	10	28	0.0000	0.0000	0.0000	4:2:0	13
00019(b)	38	11	30	0.0000	0.0000	0.0000	4:2:0	14
00022(P)	44	11	28	0.0000	0.0000	0.0000	4:2:0	14
00021(b)	42	12	30	0.0000	0.0000	0.0000	4:2:0	21
00024(P)	48	12	28	0.0000	0.0000	0.0000	4:2:0	12
00023(b)	46	13	30	0.0000	0.0000	0.0000	4:2:0	15
00026(P)	52	13	28	0.0000	0.0000	0.0000	4:2:0	14
00025(b)	50	14	30	0.0000	0.0000	0.0000	4:2:0	18
00028(P)	56	14	28	0.0000	0.0000	0.0000	4:2:0	16
00027(b)	54	15	30	0.0000	0.0000	0.0000	4:2:0	14
00030(P)	60	15	28	0.0000	0.0000	0.0000	4:2:0	15
00029(b)	58	0	30	0.0000	0.0000	0.0000	4:2:0	20
00032(P)	64	0	28	0.0000	0.0000	0.0000	4:2:0	14
00031(b)	62	1	30	0.0000	0.0000	0.0000	4:2:0	14
00034(P)	68	1	28	0.0000	0.0000	0.0000	4:2:0	16
00033(b)	66	2	30	0.0000	0.0000	0.0000	4:2:0	18
00036(P)	72	2	28	0.0000	0.0000	0.0000	4:2:0	15
00035(b)	70	3	30	0.0000	0.0000	0.0000	4:2:0	14
00038(P)	76	3	28	0.0000	0.0000	0.0000	4:2:0	17
00037(b)	74	4	30	0.0000	0.0000	0.0000	4:2:0	20
00040(P)	80	4	28	0.0000	0.0000	0.0000	4:2:0	13
00039(b)	78	5	30	0.0000	0.0000	0.0000	4:2:0	15
00042(P)	84	5	28	0.0000	0.0000	0.0000	4:2:0	16
00041(b)	82	6	30	0.0000	0.0000	0.0000	4:2:0	18
00044(P)	88	6	28	0.0000	0.0000	0.0000	4:2:0	14
00043(b)	86	7	30	0.0000	0.0000	0.0000	4:2:0	14
00046(P)	92	7	28	0.0000	0.0000	0.0000	4:2:0	16
00045(b)	90	8	30	0.0000	0.0000	0.0000	4:2:0	21
00048(P)	96	8	28	0.0000	0.0000	0.0000	4:2:0	16
00047(b)	94	9	30	0.0000	0.0000	0.0000	4:2:0	13
00050(P)	100	9	28	0.0000	0.0000	0.0000	4:2:0	16
00049(b)	98	10	30	0.0000	0.0000	0.0000	4:2:0	20
00052(P)	104	10	28	0.0000	0.0000	0.0000	4:2:0	14
00051(b)	102	11	30	0.0000	0.0000	0.0000	4:2:0	14
00054(P)	108	11	28	0.0000	0.0000	0.0000	4:2:0	14
00053(b)	106	12	30	0.0000	0.0000	0.0000	4:2:0	20
00056(P)	112	12	28	0.0000	0.0000	0.0000	4:2:0	16
00055(b)	110	13	30	0.0000	0.0000	0.0000	4:2:0	14

00058(P)	116	13	28	0.0000	0.0000	0.0000	4:2:0	15
00057(b)	114	14	30	0.0000	0.0000	0.0000	4:2:0	20
00060(P)	120	14	28	0.0000	0.0000	0.0000	4:2:0	16
00059(b)	118	15	30	0.0000	0.0000	0.0000	4:2:0	15
00062(P)	124	15	28	0.0000	0.0000	0.0000	4:2:0	14
00061(b)	122	0	30	0.0000	0.0000	0.0000	4:2:0	20
00064(P)	128	0	28	0.0000	0.0000	0.0000	4:2:0	16
00063(b)	126	1	30	0.0000	0.0000	0.0000	4:2:0	15
00066(P)	132	1	28	0.0000	0.0000	0.0000	4:2:0	15
00065(b)	130	2	30	0.0000	0.0000	0.0000	4:2:0	20
00068(P)	136	2	28	0.0000	0.0000	0.0000	4:2:0	16
00067(b)	134	3	30	0.0000	0.0000	0.0000	4:2:0	13
00070(P)	140	3	28	0.0000	0.0000	0.0000	4:2:0	15
00069(b)	138	4	30	0.0000	0.0000	0.0000	4:2:0	21
00072(P)	144	4	28	0.0000	0.0000	0.0000	4:2:0	15
00071(b)	142	5	30	0.0000	0.0000	0.0000	4:2:0	14
00074(P)	148	5	28	0.0000	0.0000	0.0000	4:2:0	14
00073(b)	146	6	30	0.0000	0.0000	0.0000	4:2:0	21
00076(P)	152	6	28	0.0000	0.0000	0.0000	4:2:0	17
00075(b)	150	7	30	0.0000	0.0000	0.0000	4:2:0	14
00078(P)	156	7	28	0.0000	0.0000	0.0000	4:2:0	14
00077(b)	154	8	30	0.0000	0.0000	0.0000	4:2:0	19
00080(P)	160	8	28	0.0000	0.0000	0.0000	4:2:0	16
00079(b)	158	9	30	0.0000	0.0000	0.0000	4:2:0	14
00082(P)	164	9	28	0.0000	0.0000	0.0000	4:2:0	13
00081(b)	162	10	30	0.0000	0.0000	0.0000	4:2:0	21
00084(P)	168	10	28	0.0000	0.0000	0.0000	4:2:0	15
00083(b)	166	11	30	0.0000	0.0000	0.0000	4:2:0	14
00086(P)	172	11	28	0.0000	0.0000	0.0000	4:2:0	13
00085(b)	170	12	30	0.0000	0.0000	0.0000	4:2:0	19
00088(P)	176	12	28	0.0000	0.0000	0.0000	4:2:0	16
00087(b)	174	13	30	0.0000	0.0000	0.0000	4:2:0	14
00090(P)	180	13	28	0.0000	0.0000	0.0000	4:2:0	15
00089(b)	178	14	30	0.0000	0.0000	0.0000	4:2:0	21
00092(P)	184	14	28	0.0000	0.0000	0.0000	4:2:0	15
00091(b)	182	15	30	0.0000	0.0000	0.0000	4:2:0	14
00094(P)	188	15	28	0.0000	0.0000	0.0000	4:2:0	14
00093(b)	186	0	30	0.0000	0.0000	0.0000	4:2:0	20
00096(P)	192	0	28	0.0000	0.0000	0.0000	4:2:0	16
00095(b)	190	1	30	0.0000	0.0000	0.0000	4:2:0	15
00098(P)	196	1	28	0.0000	0.0000	0.0000	4:2:0	14
00097(b)	194	2	30	0.0000	0.0000	0.0000	4:2:0	20
00100(P)	200	2	28	0.0000	0.0000	0.0000	4:2:0	15
00099(b)	198	3	30	0.0000	0.0000	0.0000	4:2:0	14
00102(P)	204	3	28	0.0000	0.0000	0.0000	4:2:0	14
00101(b)	202	4	30	0.0000	0.0000	0.0000	4:2:0	21
00104(P)	208	4	28	0.0000	0.0000	0.0000	4:2:0	14
00103(b)	206	5	30	0.0000	0.0000	0.0000	4:2:0	14
00106(P)	212	5	28	0.0000	0.0000	0.0000	4:2:0	15
00105(b)	210	6	30	0.0000	0.0000	0.0000	4:2:0	22
00108(P)	216	6	28	0.0000	0.0000	0.0000	4:2:0	17
00107(b)	214	7	30	0.0000	0.0000	0.0000	4:2:0	14
00110(P)	220	7	28	0.0000	0.0000	0.0000	4:2:0	14
00109(b)	218	8	30	0.0000	0.0000	0.0000	4:2:0	22
00112(P)	224	8	28	0.0000	0.0000	0.0000	4:2:0	14
00111(b)	222	9	30	0.0000	0.0000	0.0000	4:2:0	14
00114(P)	228	9	28	0.0000	0.0000	0.0000	4:2:0	16
00113(b)	226	10	30	0.0000	0.0000	0.0000	4:2:0	19
00116(P)	232	10	28	0.0000	0.0000	0.0000	4:2:0	16
00115(b)	230	11	30	0.0000	0.0000	0.0000	4:2:0	14

00118(P)	236	11	28	0.0000	0.0000	0.0000	4:2:0	13
00117(b)	234	12	30	0.0000	0.0000	0.0000	4:2:0	19
00120(P)	240	12	28	0.0000	0.0000	0.0000	4:2:0	16
00119(b)	238	13	30	0.0000	0.0000	0.0000	4:2:0	15
00122(P)	244	13	28	0.0000	0.0000	0.0000	4:2:0	14
00121(b)	242	14	30	0.0000	0.0000	0.0000	4:2:0	23
00124(P)	248	14	28	0.0000	0.0000	0.0000	4:2:0	16
00123(b)	246	15	30	0.0000	0.0000	0.0000	4:2:0	14
00126(P)	252	15	28	0.0000	0.0000	0.0000	4:2:0	15
00125(b)	250	0	30	0.0000	0.0000	0.0000	4:2:0	18
00128(P)	256	0	28	0.0000	0.0000	0.0000	4:2:0	17
00127(b)	254	1	30	0.0000	0.0000	0.0000	4:2:0	15
00130(P)	260	1	28	0.0000	0.0000	0.0000	4:2:0	15
00129(b)	258	2	30	0.0000	0.0000	0.0000	4:2:0	20
00132(P)	264	2	28	0.0000	0.0000	0.0000	4:2:0	17
00131(b)	262	3	30	0.0000	0.0000	0.0000	4:2:0	15
00134(P)	268	3	28	0.0000	0.0000	0.0000	4:2:0	15
00133(b)	266	4	30	0.0000	0.0000	0.0000	4:2:0	22
00136(P)	272	4	28	0.0000	0.0000	0.0000	4:2:0	15
00135(b)	270	5	30	0.0000	0.0000	0.0000	4:2:0	15
00138(P)	276	5	28	0.0000	0.0000	0.0000	4:2:0	15
00137(b)	274	6	30	0.0000	0.0000	0.0000	4:2:0	22
00140(P)	280	6	28	0.0000	0.0000	0.0000	4:2:0	15
00139(b)	278	7	30	0.0000	0.0000	0.0000	4:2:0	16
00142(P)	284	7	28	0.0000	0.0000	0.0000	4:2:0	14
00141(b)	282	8	30	0.0000	0.0000	0.0000	4:2:0	19
00144(P)	288	8	28	0.0000	0.0000	0.0000	4:2:0	16
00143(b)	286	9	30	0.0000	0.0000	0.0000	4:2:0	15
00146(P)	292	9	28	0.0000	0.0000	0.0000	4:2:0	14
00145(b)	290	10	30	0.0000	0.0000	0.0000	4:2:0	19
00148(P)	296	10	28	0.0000	0.0000	0.0000	4:2:0	16
00147(b)	294	11	30	0.0000	0.0000	0.0000	4:2:0	15
00150(P)	300	11	28	0.0000	0.0000	0.0000	4:2:0	15
00149(b)	298	12	30	0.0000	0.0000	0.0000	4:2:0	21
00152(P)	304	12	28	0.0000	0.0000	0.0000	4:2:0	16
00151(b)	302	13	30	0.0000	0.0000	0.0000	4:2:0	14
00154(P)	308	13	28	0.0000	0.0000	0.0000	4:2:0	15
00153(b)	306	14	30	0.0000	0.0000	0.0000	4:2:0	21
00156(P)	312	14	28	0.0000	0.0000	0.0000	4:2:0	17
00155(b)	310	15	30	0.0000	0.0000	0.0000	4:2:0	15
00158(P)	316	15	28	0.0000	0.0000	0.0000	4:2:0	14
00157(b)	314	0	30	0.0000	0.0000	0.0000	4:2:0	18
00160(P)	320	0	28	0.0000	0.0000	0.0000	4:2:0	17
00159(b)	318	1	30	0.0000	0.0000	0.0000	4:2:0	15
00162(P)	324	1	28	0.0000	0.0000	0.0000	4:2:0	14
00161(b)	322	2	30	0.0000	0.0000	0.0000	4:2:0	18
00164(P)	328	2	28	0.0000	0.0000	0.0000	4:2:0	17
00163(b)	326	3	30	0.0000	0.0000	0.0000	4:2:0	17
00166(P)	332	3	28	0.0000	0.0000	0.0000	4:2:0	15
00165(b)	330	4	30	0.0000	0.0000	0.0000	4:2:0	19
00168(P)	336	4	28	0.0000	0.0000	0.0000	4:2:0	16
00167(b)	334	5	30	0.0000	0.0000	0.0000	4:2:0	16
00170(P)	340	5	28	0.0000	0.0000	0.0000	4:2:0	15
00169(b)	338	6	30	0.0000	0.0000	0.0000	4:2:0	22
00172(P)	344	6	28	0.0000	0.0000	0.0000	4:2:0	17
00171(b)	342	7	30	0.0000	0.0000	0.0000	4:2:0	15
00174(P)	348	7	28	0.0000	0.0000	0.0000	4:2:0	16
00173(b)	346	8	30	0.0000	0.0000	0.0000	4:2:0	18
00176(P)	352	8	28	0.0000	0.0000	0.0000	4:2:0	17
00175(b)	350	9	30	0.0000	0.0000	0.0000	4:2:0	15

00178(P)	356	9	28	0.0000	0.0000	0.0000	4:2:0	17
00177(b)	354	10	30	0.0000	0.0000	0.0000	4:2:0	23
00180(P)	360	10	28	0.0000	0.0000	0.0000	4:2:0	17
00179(b)	358	11	30	0.0000	0.0000	0.0000	4:2:0	15
00182(P)	364	11	28	0.0000	0.0000	0.0000	4:2:0	16
00181(b)	362	12	30	0.0000	0.0000	0.0000	4:2:0	22
00184(P)	368	12	28	0.0000	0.0000	0.0000	4:2:0	17
00183(b)	366	13	30	0.0000	0.0000	0.0000	4:2:0	15
00186(P)	372	13	28	0.0000	0.0000	0.0000	4:2:0	17
00185(b)	370	14	30	0.0000	0.0000	0.0000	4:2:0	19
00188(P)	376	14	28	0.0000	0.0000	0.0000	4:2:0	19
00187(b)	374	15	30	0.0000	0.0000	0.0000	4:2:0	13
00190(P)	380	15	28	0.0000	0.0000	0.0000	4:2:0	18
00189(b)	378	0	30	0.0000	0.0000	0.0000	4:2:0	22
00192(P)	384	0	28	0.0000	0.0000	0.0000	4:2:0	17
00191(b)	382	1	30	0.0000	0.0000	0.0000	4:2:0	13
00194(P)	388	1	28	0.0000	0.0000	0.0000	4:2:0	16
00193(b)	386	2	30	0.0000	0.0000	0.0000	4:2:0	20
00196(P)	392	2	28	0.0000	0.0000	0.0000	4:2:0	18
00195(b)	390	3	30	0.0000	0.0000	0.0000	4:2:0	13
00198(P)	396	3	28	0.0000	0.0000	0.0000	4:2:0	16
00197(b)	394	4	30	0.0000	0.0000	0.0000	4:2:0	21
00200(P)	400	4	28	0.0000	0.0000	0.0000	4:2:0	18
00199(b)	398	5	30	0.0000	0.0000	0.0000	4:2:0	14
00202(P)	404	5	28	0.0000	0.0000	0.0000	4:2:0	16
00201(b)	402	6	30	0.0000	0.0000	0.0000	4:2:0	24
00204(P)	408	6	28	0.0000	0.0000	0.0000	4:2:0	18
00203(b)	406	7	30	0.0000	0.0000	0.0000	4:2:0	13
00206(P)	412	7	28	0.0000	0.0000	0.0000	4:2:0	14
00205(b)	410	8	30	0.0000	0.0000	0.0000	4:2:0	19
00208(P)	416	8	28	0.0000	0.0000	0.0000	4:2:0	18
00207(b)	414	9	30	0.0000	0.0000	0.0000	4:2:0	13
00210(P)	420	9	28	0.0000	0.0000	0.0000	4:2:0	16
00209(b)	418	10	30	0.0000	0.0000	0.0000	4:2:0	23
00212(P)	424	10	28	0.0000	0.0000	0.0000	4:2:0	17
00211(b)	422	11	30	0.0000	0.0000	0.0000	4:2:0	14
00214(P)	428	11	28	0.0000	0.0000	0.0000	4:2:0	15
00213(b)	426	12	30	0.0000	0.0000	0.0000	4:2:0	20
00216(P)	432	12	28	0.0000	0.0000	0.0000	4:2:0	16
00215(b)	430	13	30	0.0000	0.0000	0.0000	4:2:0	13
00218(P)	436	13	28	0.0000	0.0000	0.0000	4:2:0	15
00217(b)	434	14	30	0.0000	0.0000	0.0000	4:2:0	20
00220(P)	440	14	28	0.0000	0.0000	0.0000	4:2:0	18
00219(b)	438	15	30	0.0000	0.0000	0.0000	4:2:0	13
00222(P)	444	15	28	0.0000	0.0000	0.0000	4:2:0	13
00221(b)	442	0	30	0.0000	0.0000	0.0000	4:2:0	24
00224(P)	448	0	28	0.0000	0.0000	0.0000	4:2:0	17
00223(b)	446	1	30	0.0000	0.0000	0.0000	4:2:0	14
00226(P)	452	1	28	0.0000	0.0000	0.0000	4:2:0	15
00225(b)	450	2	30	0.0000	0.0000	0.0000	4:2:0	23
00228(P)	456	2	28	0.0000	0.0000	0.0000	4:2:0	18
00227(b)	454	3	30	0.0000	0.0000	0.0000	4:2:0	13
00230(P)	460	3	28	0.0000	0.0000	0.0000	4:2:0	15
00229(b)	458	4	30	0.0000	0.0000	0.0000	4:2:0	19
00232(P)	464	4	28	0.0000	0.0000	0.0000	4:2:0	18
00231(b)	462	5	30	0.0000	0.0000	0.0000	4:2:0	13
00234(P)	468	5	28	0.0000	0.0000	0.0000	4:2:0	15
00233(b)	466	6	30	0.0000	0.0000	0.0000	4:2:0	23
00236(P)	472	6	28	0.0000	0.0000	0.0000	4:2:0	17
00235(b)	470	7	30	0.0000	0.0000	0.0000	4:2:0	14

00238(P)	476	7	28	0.0000	0.0000	0.0000	4:2:0	15
00237(b)	474	8	30	0.0000	0.0000	0.0000	4:2:0	22
00240(P)	480	8	28	0.0000	0.0000	0.0000	4:2:0	18
00239(b)	478	9	30	0.0000	0.0000	0.0000	4:2:0	14
00242(P)	484	9	28	0.0000	0.0000	0.0000	4:2:0	16
00241(b)	482	10	30	0.0000	0.0000	0.0000	4:2:0	19
00244(P)	488	10	28	0.0000	0.0000	0.0000	4:2:0	18
00243(b)	486	11	30	0.0000	0.0000	0.0000	4:2:0	15
00246(P)	492	11	28	0.0000	0.0000	0.0000	4:2:0	14
00245(b)	490	12	30	0.0000	0.0000	0.0000	4:2:0	22
00248(P)	496	12	28	0.0000	0.0000	0.0000	4:2:0	17
00247(b)	494	13	30	0.0000	0.0000	0.0000	4:2:0	15
00250(P)	500	13	28	0.0000	0.0000	0.0000	4:2:0	15
00249(b)	498	14	30	0.0000	0.0000	0.0000	4:2:0	21
00252(P)	504	14	28	0.0000	0.0000	0.0000	4:2:0	17
00251(b)	502	15	30	0.0000	0.0000	0.0000	4:2:0	16
00254(P)	508	15	28	0.0000	0.0000	0.0000	4:2:0	15
00253(b)	506	0	30	0.0000	0.0000	0.0000	4:2:0	20
00256(P)	512	0	28	0.0000	0.0000	0.0000	4:2:0	16
00255(b)	510	1	30	0.0000	0.0000	0.0000	4:2:0	16
00258(P)	516	1	28	0.0000	0.0000	0.0000	4:2:0	14
00257(b)	514	2	30	0.0000	0.0000	0.0000	4:2:0	21
00260(P)	520	2	28	0.0000	0.0000	0.0000	4:2:0	16
00259(b)	518	3	30	0.0000	0.0000	0.0000	4:2:0	14
00262(P)	524	3	28	0.0000	0.0000	0.0000	4:2:0	15
00261(b)	522	4	30	0.0000	0.0000	0.0000	4:2:0	21
00264(P)	528	4	28	0.0000	0.0000	0.0000	4:2:0	17
00263(b)	526	5	30	0.0000	0.0000	0.0000	4:2:0	16
00266(P)	532	5	28	0.0000	0.0000	0.0000	4:2:0	18
00265(b)	530	6	30	0.0000	0.0000	0.0000	4:2:0	22
00268(P)	536	6	28	0.0000	0.0000	0.0000	4:2:0	17
00267(b)	534	7	30	0.0000	0.0000	0.0000	4:2:0	15
00270(P)	540	7	28	0.0000	0.0000	0.0000	4:2:0	17
00269(b)	538	8	30	0.0000	0.0000	0.0000	4:2:0	22
00272(P)	544	8	28	0.0000	0.0000	0.0000	4:2:0	17
00271(b)	542	9	30	0.0000	0.0000	0.0000	4:2:0	15
00274(P)	548	9	28	0.0000	0.0000	0.0000	4:2:0	15
00273(b)	546	10	30	0.0000	0.0000	0.0000	4:2:0	22
00276(P)	552	10	28	0.0000	0.0000	0.0000	4:2:0	16
00275(b)	550	11	30	0.0000	0.0000	0.0000	4:2:0	17
00278(P)	556	11	28	0.0000	0.0000	0.0000	4:2:0	15
00277(b)	554	12	30	0.0000	0.0000	0.0000	4:2:0	23
00280(P)	560	12	28	0.0000	0.0000	0.0000	4:2:0	15
00279(b)	558	13	30	0.0000	0.0000	0.0000	4:2:0	15
00282(P)	564	13	28	0.0000	0.0000	0.0000	4:2:0	14
00281(b)	562	14	30	0.0000	0.0000	0.0000	4:2:0	21
00284(P)	568	14	28	0.0000	0.0000	0.0000	4:2:0	16
00283(b)	566	15	30	0.0000	0.0000	0.0000	4:2:0	15
00286(P)	572	15	28	0.0000	0.0000	0.0000	4:2:0	15
00285(b)	570	0	30	0.0000	0.0000	0.0000	4:2:0	22
00288(P)	576	0	28	0.0000	0.0000	0.0000	4:2:0	16
00287(b)	574	1	30	0.0000	0.0000	0.0000	4:2:0	16
00290(P)	580	1	28	0.0000	0.0000	0.0000	4:2:0	16
00289(b)	578	2	30	0.0000	0.0000	0.0000	4:2:0	21
00292(P)	584	2	28	0.0000	0.0000	0.0000	4:2:0	17
00291(b)	582	3	30	0.0000	0.0000	0.0000	4:2:0	15
00294(P)	588	3	28	0.0000	0.0000	0.0000	4:2:0	17
00293(b)	586	4	30	0.0000	0.0000	0.0000	4:2:0	20
00296(P)	592	4	28	0.0000	0.0000	0.0000	4:2:0	18
00295(b)	590	5	30	0.0000	0.0000	0.0000	4:2:0	14

```
00298( P )      596     5    28    0.0000    0.0000    0.0000   4:2:0       15
end decoding
------------------- Average SNR all frames -----------------------------------
 SNR Y(dB)           :  0.00
 SNR U(dB)           :  0.00
 SNR V(dB)           :  0.00
 Total decoding time : 4.924 sec (60.723 fps)
------------------------------------------------------------------------------
 Exit JM 13 (FREXt) decoder, ver 13.0
```

A.4 Simulation result of the SpecC clean exploration model

```
------------------------------ JM 13.0 (FREXt) -------------------------------
 Decoder config file                    : decoder.cfg
------------------------------------------------------------------------------
 Input H.264 bitstream                  : test.264
 Output decoded YUV                     : test_dec.yuv
 Output status file                     : log.dec
 Input reference file                   : test_rec.yuv
------------------------------------------------------------------------------
POC must = frame# or field# for SNRs to be correct
------------------------------------------------------------------------------
  Frame          POC   Pic#   QP     SnrY      SnrU      SnrV    Y:U:V Time(ms)
------------------------------------------------------------------------------
00000(IDR)         0      0   28   0.0000    0.0000    0.0000   4:2:0       12
00002( P )         4      1   28   0.0000    0.0000    0.0000   4:2:0       13
00001( b )         2      2   30   0.0000    0.0000    0.0000   4:2:0       14
00004( P )         8      2   28   0.0000    0.0000    0.0000   4:2:0       11
00003( b )         6      3   30   0.0000    0.0000    0.0000   4:2:0       10
00006( P )        12      3   28   0.0000    0.0000    0.0000   4:2:0       13
00005( b )        10      4   30   0.0000    0.0000    0.0000   4:2:0       15
00008( P )        16      4   28   0.0000    0.0000    0.0000   4:2:0       11
00007( b )        14      5   30   0.0000    0.0000    0.0000   4:2:0       11
00010( P )        20      5   28   0.0000    0.0000    0.0000   4:2:0       12
00009( b )        18      6   30   0.0000    0.0000    0.0000   4:2:0       16
00012( P )        24      6   28   0.0000    0.0000    0.0000   4:2:0       10
00011( b )        22      7   30   0.0000    0.0000    0.0000   4:2:0       10
00014( P )        28      7   28   0.0000    0.0000    0.0000   4:2:0       12
00013( b )        26      8   30   0.0000    0.0000    0.0000   4:2:0       15
00016( P )        32      8   28   0.0000    0.0000    0.0000   4:2:0       12
00015( b )        30      9   30   0.0000    0.0000    0.0000   4:2:0       11
00018( P )        36      9   28   0.0000    0.0000    0.0000   4:2:0       13
00017( b )        34     10   30   0.0000    0.0000    0.0000   4:2:0       15
00020( P )        40     10   28   0.0000    0.0000    0.0000   4:2:0       12
00019( b )        38     11   30   0.0000    0.0000    0.0000   4:2:0       11
00022( P )        44     11   28   0.0000    0.0000    0.0000   4:2:0       12
00021( b )        42     12   30   0.0000    0.0000    0.0000   4:2:0       16
00024( P )        48     12   28   0.0000    0.0000    0.0000   4:2:0       11
00023( b )        46     13   30   0.0000    0.0000    0.0000   4:2:0       11
00026( P )        52     13   28   0.0000    0.0000    0.0000   4:2:0       13
00025( b )        50     14   30   0.0000    0.0000    0.0000   4:2:0       14
00028( P )        56     14   28   0.0000    0.0000    0.0000   4:2:0       14
00027( b )        54     15   30   0.0000    0.0000    0.0000   4:2:0       12
00030( P )        60     15   28   0.0000    0.0000    0.0000   4:2:0       13
00029( b )        58      0   30   0.0000    0.0000    0.0000   4:2:0       15
00032( P )        64      0   28   0.0000    0.0000    0.0000   4:2:0       13
00031( b )        62      1   30   0.0000    0.0000    0.0000   4:2:0       11
00034( P )        68      1   28   0.0000    0.0000    0.0000   4:2:0       13
00033( b )        66      2   30   0.0000    0.0000    0.0000   4:2:0       15
00036( P )        72      2   28   0.0000    0.0000    0.0000   4:2:0       14
```

00035(b)	70	3	30	0.0000	0.0000	0.0000	4:2:0	11
00038(P)	76	3	28	0.0000	0.0000	0.0000	4:2:0	13
00037(b)	74	4	30	0.0000	0.0000	0.0000	4:2:0	16
00040(P)	80	4	28	0.0000	0.0000	0.0000	4:2:0	12
00039(b)	78	5	30	0.0000	0.0000	0.0000	4:2:0	12
00042(P)	84	5	28	0.0000	0.0000	0.0000	4:2:0	13
00041(b)	82	6	30	0.0000	0.0000	0.0000	4:2:0	15
00044(P)	88	6	28	0.0000	0.0000	0.0000	4:2:0	12
00043(b)	86	7	30	0.0000	0.0000	0.0000	4:2:0	11
00046(P)	92	7	28	0.0000	0.0000	0.0000	4:2:0	13
00045(b)	90	8	30	0.0000	0.0000	0.0000	4:2:0	16
00048(P)	96	8	28	0.0000	0.0000	0.0000	4:2:0	15
00047(b)	94	9	30	0.0000	0.0000	0.0000	4:2:0	10
00050(P)	100	9	28	0.0000	0.0000	0.0000	4:2:0	14
00049(b)	98	10	30	0.0000	0.0000	0.0000	4:2:0	15
00052(P)	104	10	28	0.0000	0.0000	0.0000	4:2:0	13
00051(b)	102	11	30	0.0000	0.0000	0.0000	4:2:0	11
00054(P)	108	11	28	0.0000	0.0000	0.0000	4:2:0	13
00053(b)	106	12	30	0.0000	0.0000	0.0000	4:2:0	15
00056(P)	112	12	28	0.0000	0.0000	0.0000	4:2:0	14
00055(b)	110	13	30	0.0000	0.0000	0.0000	4:2:0	11
00058(P)	116	13	28	0.0000	0.0000	0.0000	4:2:0	13
00057(b)	114	14	30	0.0000	0.0000	0.0000	4:2:0	15
00060(P)	120	14	28	0.0000	0.0000	0.0000	4:2:0	15
00059(b)	118	15	30	0.0000	0.0000	0.0000	4:2:0	11
00062(P)	124	15	28	0.0000	0.0000	0.0000	4:2:0	13
00061(b)	122	0	30	0.0000	0.0000	0.0000	4:2:0	16
00064(P)	128	0	28	0.0000	0.0000	0.0000	4:2:0	14
00063(b)	126	1	30	0.0000	0.0000	0.0000	4:2:0	11
00066(P)	132	1	28	0.0000	0.0000	0.0000	4:2:0	13
00065(b)	130	2	30	0.0000	0.0000	0.0000	4:2:0	16
00068(P)	136	2	28	0.0000	0.0000	0.0000	4:2:0	13
00067(b)	134	3	30	0.0000	0.0000	0.0000	4:2:0	11
00070(P)	140	3	28	0.0000	0.0000	0.0000	4:2:0	13
00069(b)	138	4	30	0.0000	0.0000	0.0000	4:2:0	16
00072(P)	144	4	28	0.0000	0.0000	0.0000	4:2:0	14
00071(b)	142	5	30	0.0000	0.0000	0.0000	4:2:0	11
00074(P)	148	5	28	0.0000	0.0000	0.0000	4:2:0	12
00073(b)	146	6	30	0.0000	0.0000	0.0000	4:2:0	16
00076(P)	152	6	28	0.0000	0.0000	0.0000	4:2:0	14
00075(b)	150	7	30	0.0000	0.0000	0.0000	4:2:0	11
00078(P)	156	7	28	0.0000	0.0000	0.0000	4:2:0	13
00077(b)	154	8	30	0.0000	0.0000	0.0000	4:2:0	15
00080(P)	160	8	28	0.0000	0.0000	0.0000	4:2:0	15
00079(b)	158	9	30	0.0000	0.0000	0.0000	4:2:0	11
00082(P)	164	9	28	0.0000	0.0000	0.0000	4:2:0	12
00081(b)	162	10	30	0.0000	0.0000	0.0000	4:2:0	16
00084(P)	168	10	28	0.0000	0.0000	0.0000	4:2:0	13
00083(b)	166	11	30	0.0000	0.0000	0.0000	4:2:0	11
00086(P)	172	11	28	0.0000	0.0000	0.0000	4:2:0	13
00085(b)	170	12	30	0.0000	0.0000	0.0000	4:2:0	15
00088(P)	176	12	28	0.0000	0.0000	0.0000	4:2:0	14
00087(b)	174	13	30	0.0000	0.0000	0.0000	4:2:0	12
00090(P)	180	13	28	0.0000	0.0000	0.0000	4:2:0	12
00089(b)	178	14	30	0.0000	0.0000	0.0000	4:2:0	17
00092(P)	184	14	28	0.0000	0.0000	0.0000	4:2:0	13
00091(b)	182	15	30	0.0000	0.0000	0.0000	4:2:0	11
00094(P)	188	15	28	0.0000	0.0000	0.0000	4:2:0	12
00093(b)	186	0	30	0.0000	0.0000	0.0000	4:2:0	16
00096(P)	192	0	28	0.0000	0.0000	0.0000	4:2:0	14

00095(b)	190	1	30	0.0000	0.0000	0.0000	4:2:0	12
00098(P)	196	1	28	0.0000	0.0000	0.0000	4:2:0	13
00097(b)	194	2	30	0.0000	0.0000	0.0000	4:2:0	16
00100(P)	200	2	28	0.0000	0.0000	0.0000	4:2:0	13
00099(b)	198	3	30	0.0000	0.0000	0.0000	4:2:0	12
00102(P)	204	3	28	0.0000	0.0000	0.0000	4:2:0	13
00101(b)	202	4	30	0.0000	0.0000	0.0000	4:2:0	16
00104(P)	208	4	28	0.0000	0.0000	0.0000	4:2:0	12
00103(b)	206	5	30	0.0000	0.0000	0.0000	4:2:0	11
00106(P)	212	5	28	0.0000	0.0000	0.0000	4:2:0	13
00105(b)	210	6	30	0.0000	0.0000	0.0000	4:2:0	16
00108(P)	216	6	28	0.0000	0.0000	0.0000	4:2:0	15
00107(b)	214	7	30	0.0000	0.0000	0.0000	4:2:0	11
00110(P)	220	7	28	0.0000	0.0000	0.0000	4:2:0	13
00109(b)	218	8	30	0.0000	0.0000	0.0000	4:2:0	16
00112(P)	224	8	28	0.0000	0.0000	0.0000	4:2:0	13
00111(b)	222	9	30	0.0000	0.0000	0.0000	4:2:0	11
00114(P)	228	9	28	0.0000	0.0000	0.0000	4:2:0	13
00113(b)	226	10	30	0.0000	0.0000	0.0000	4:2:0	16
00116(P)	232	10	28	0.0000	0.0000	0.0000	4:2:0	14
00115(b)	230	11	30	0.0000	0.0000	0.0000	4:2:0	12
00118(P)	236	11	28	0.0000	0.0000	0.0000	4:2:0	11
00117(b)	234	12	30	0.0000	0.0000	0.0000	4:2:0	16
00120(P)	240	12	28	0.0000	0.0000	0.0000	4:2:0	14
00119(b)	238	13	30	0.0000	0.0000	0.0000	4:2:0	12
00122(P)	244	13	28	0.0000	0.0000	0.0000	4:2:0	12
00121(b)	242	14	30	0.0000	0.0000	0.0000	4:2:0	17
00124(P)	248	14	28	0.0000	0.0000	0.0000	4:2:0	15
00123(b)	246	15	30	0.0000	0.0000	0.0000	4:2:0	11
00126(P)	252	15	28	0.0000	0.0000	0.0000	4:2:0	13
00125(b)	250	0	30	0.0000	0.0000	0.0000	4:2:0	15
00128(P)	256	0	28	0.0000	0.0000	0.0000	4:2:0	15
00127(b)	254	1	30	0.0000	0.0000	0.0000	4:2:0	12
00130(P)	260	1	28	0.0000	0.0000	0.0000	4:2:0	13
00129(b)	258	2	30	0.0000	0.0000	0.0000	4:2:0	16
00132(P)	264	2	28	0.0000	0.0000	0.0000	4:2:0	14
00131(b)	262	3	30	0.0000	0.0000	0.0000	4:2:0	12
00134(P)	268	3	28	0.0000	0.0000	0.0000	4:2:0	12
00133(b)	266	4	30	0.0000	0.0000	0.0000	4:2:0	17
00136(P)	272	4	28	0.0000	0.0000	0.0000	4:2:0	14
00135(b)	270	5	30	0.0000	0.0000	0.0000	4:2:0	12
00138(P)	276	5	28	0.0000	0.0000	0.0000	4:2:0	12
00137(b)	274	6	30	0.0000	0.0000	0.0000	4:2:0	17
00140(P)	280	6	28	0.0000	0.0000	0.0000	4:2:0	15
00139(b)	278	7	30	0.0000	0.0000	0.0000	4:2:0	12
00142(P)	284	7	28	0.0000	0.0000	0.0000	4:2:0	12
00141(b)	282	8	30	0.0000	0.0000	0.0000	4:2:0	16
00144(P)	288	8	28	0.0000	0.0000	0.0000	4:2:0	14
00143(b)	286	9	30	0.0000	0.0000	0.0000	4:2:0	12
00146(P)	292	9	28	0.0000	0.0000	0.0000	4:2:0	13
00145(b)	290	10	30	0.0000	0.0000	0.0000	4:2:0	15
00148(P)	296	10	28	0.0000	0.0000	0.0000	4:2:0	15
00147(b)	294	11	30	0.0000	0.0000	0.0000	4:2:0	12
00150(P)	300	11	28	0.0000	0.0000	0.0000	4:2:0	13
00149(b)	298	12	30	0.0000	0.0000	0.0000	4:2:0	16
00152(P)	304	12	28	0.0000	0.0000	0.0000	4:2:0	15
00151(b)	302	13	30	0.0000	0.0000	0.0000	4:2:0	11
00154(P)	308	13	28	0.0000	0.0000	0.0000	4:2:0	14
00153(b)	306	14	30	0.0000	0.0000	0.0000	4:2:0	16
00156(P)	312	14	28	0.0000	0.0000	0.0000	4:2:0	15

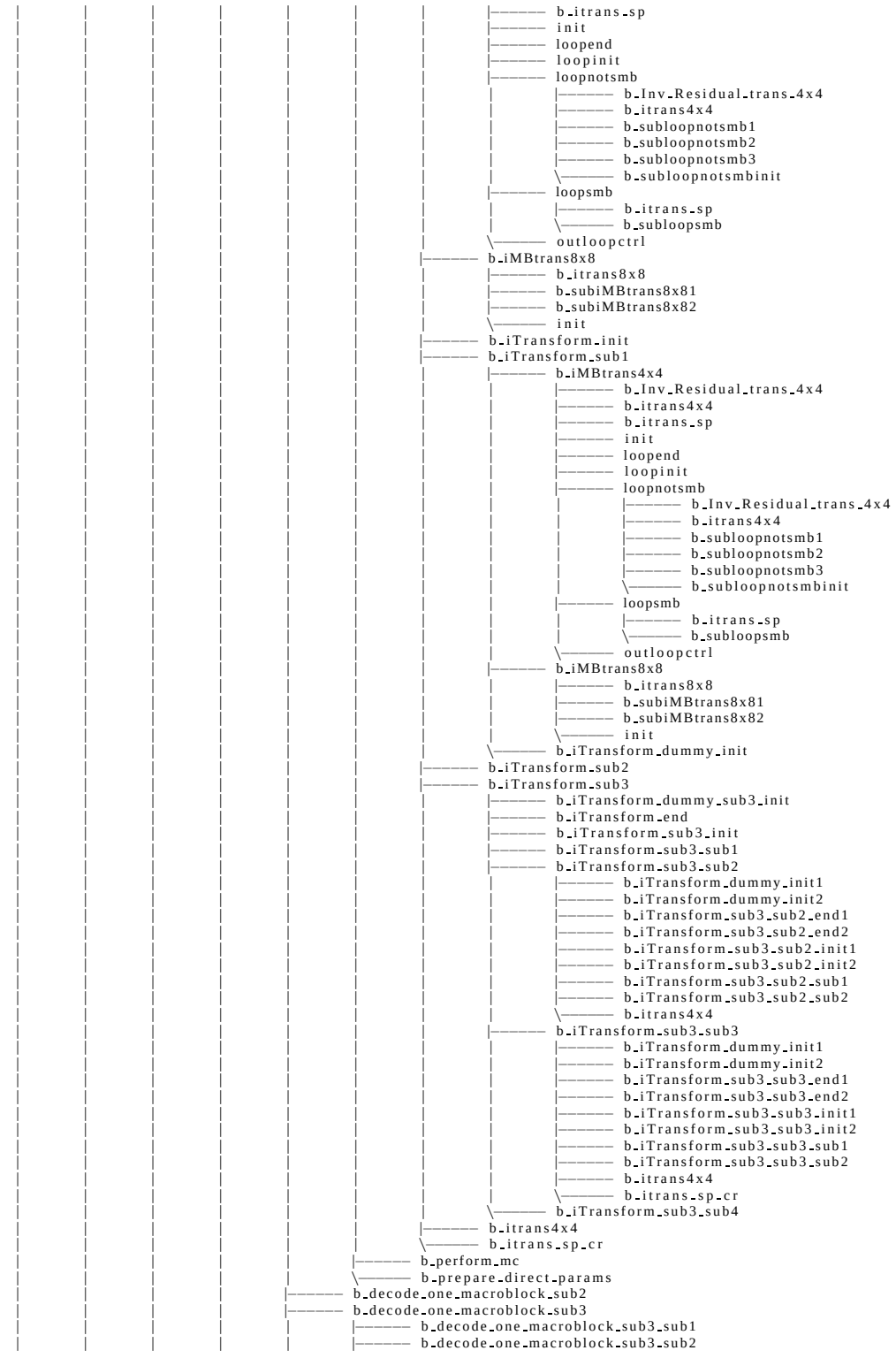
00155(b)	310	15	30	0.0000	0.0000	0.0000	4:2:0	12
00158(P)	316	15	28	0.0000	0.0000	0.0000	4:2:0	13
00157(b)	314	0	30	0.0000	0.0000	0.0000	4:2:0	15
00160(P)	320	0	28	0.0000	0.0000	0.0000	4:2:0	15
00159(b)	318	1	30	0.0000	0.0000	0.0000	4:2:0	12
00162(P)	324	1	28	0.0000	0.0000	0.0000	4:2:0	12
00161(b)	322	2	30	0.0000	0.0000	0.0000	4:2:0	15
00164(P)	328	2	28	0.0000	0.0000	0.0000	4:2:0	15
00163(b)	326	3	30	0.0000	0.0000	0.0000	4:2:0	13
00166(P)	332	3	28	0.0000	0.0000	0.0000	4:2:0	13
00165(b)	330	4	30	0.0000	0.0000	0.0000	4:2:0	16
00168(P)	336	4	28	0.0000	0.0000	0.0000	4:2:0	14
00167(b)	334	5	30	0.0000	0.0000	0.0000	4:2:0	12
00170(P)	340	5	28	0.0000	0.0000	0.0000	4:2:0	13
00169(b)	338	6	30	0.0000	0.0000	0.0000	4:2:0	17
00172(P)	344	6	28	0.0000	0.0000	0.0000	4:2:0	16
00171(b)	342	7	30	0.0000	0.0000	0.0000	4:2:0	12
00174(P)	348	7	28	0.0000	0.0000	0.0000	4:2:0	14
00173(b)	346	8	30	0.0000	0.0000	0.0000	4:2:0	15
00176(P)	352	8	28	0.0000	0.0000	0.0000	4:2:0	16
00175(b)	350	9	30	0.0000	0.0000	0.0000	4:2:0	13
00178(P)	356	9	28	0.0000	0.0000	0.0000	4:2:0	14
00177(b)	354	10	30	0.0000	0.0000	0.0000	4:2:0	17
00180(P)	360	10	28	0.0000	0.0000	0.0000	4:2:0	16
00179(b)	358	11	30	0.0000	0.0000	0.0000	4:2:0	12
00182(P)	364	11	28	0.0000	0.0000	0.0000	4:2:0	14
00181(b)	362	12	30	0.0000	0.0000	0.0000	4:2:0	17
00184(P)	368	12	28	0.0000	0.0000	0.0000	4:2:0	16
00183(b)	366	13	30	0.0000	0.0000	0.0000	4:2:0	12
00186(P)	372	13	28	0.0000	0.0000	0.0000	4:2:0	14
00185(b)	370	14	30	0.0000	0.0000	0.0000	4:2:0	16
00188(P)	376	14	28	0.0000	0.0000	0.0000	4:2:0	17
00187(b)	374	15	30	0.0000	0.0000	0.0000	4:2:0	11
00190(P)	380	15	28	0.0000	0.0000	0.0000	4:2:0	15
00189(b)	378	0	30	0.0000	0.0000	0.0000	4:2:0	18
00192(P)	384	0	28	0.0000	0.0000	0.0000	4:2:0	16
00191(b)	382	1	30	0.0000	0.0000	0.0000	4:2:0	11
00194(P)	388	1	28	0.0000	0.0000	0.0000	4:2:0	14
00193(b)	386	2	30	0.0000	0.0000	0.0000	4:2:0	16
00196(P)	392	2	28	0.0000	0.0000	0.0000	4:2:0	16
00195(b)	390	3	30	0.0000	0.0000	0.0000	4:2:0	11
00198(P)	396	3	28	0.0000	0.0000	0.0000	4:2:0	13
00197(b)	394	4	30	0.0000	0.0000	0.0000	4:2:0	18
00200(P)	400	4	28	0.0000	0.0000	0.0000	4:2:0	16
00199(b)	398	5	30	0.0000	0.0000	0.0000	4:2:0	11
00202(P)	404	5	28	0.0000	0.0000	0.0000	4:2:0	14
00201(b)	402	6	30	0.0000	0.0000	0.0000	4:2:0	18
00204(P)	408	6	28	0.0000	0.0000	0.0000	4:2:0	16
00203(b)	406	7	30	0.0000	0.0000	0.0000	4:2:0	11
00206(P)	412	7	28	0.0000	0.0000	0.0000	4:2:0	13
00205(b)	410	8	30	0.0000	0.0000	0.0000	4:2:0	16
00208(P)	416	8	28	0.0000	0.0000	0.0000	4:2:0	16
00207(b)	414	9	30	0.0000	0.0000	0.0000	4:2:0	11
00210(P)	420	9	28	0.0000	0.0000	0.0000	4:2:0	14
00209(b)	418	10	30	0.0000	0.0000	0.0000	4:2:0	18
00212(P)	424	10	28	0.0000	0.0000	0.0000	4:2:0	16
00211(b)	422	11	30	0.0000	0.0000	0.0000	4:2:0	11
00214(P)	428	11	28	0.0000	0.0000	0.0000	4:2:0	13
00213(b)	426	12	30	0.0000	0.0000	0.0000	4:2:0	17
00216(P)	432	12	28	0.0000	0.0000	0.0000	4:2:0	14

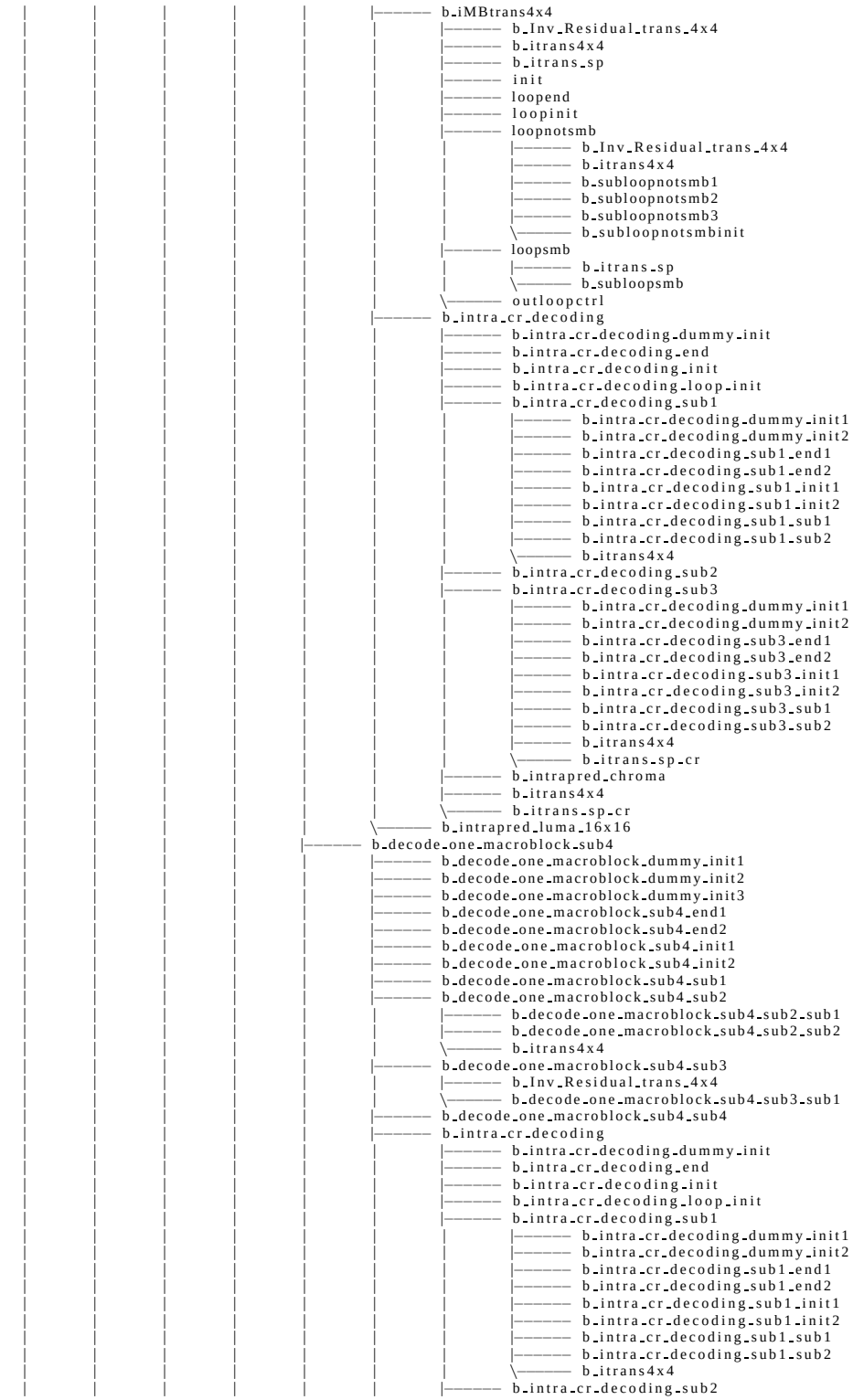
00215(b)	430	13	30	0.0000	0.0000	0.0000	4:2:0	11
00218(P)	436	13	28	0.0000	0.0000	0.0000	4:2:0	13
00217(b)	434	14	30	0.0000	0.0000	0.0000	4:2:0	17
00220(P)	440	14	28	0.0000	0.0000	0.0000	4:2:0	15
00219(b)	438	15	30	0.0000	0.0000	0.0000	4:2:0	11
00222(P)	444	15	28	0.0000	0.0000	0.0000	4:2:0	12
00221(b)	442	0	30	0.0000	0.0000	0.0000	4:2:0	19
00224(P)	448	0	28	0.0000	0.0000	0.0000	4:2:0	15
00223(b)	446	1	30	0.0000	0.0000	0.0000	4:2:0	11
00226(P)	452	1	28	0.0000	0.0000	0.0000	4:2:0	13
00225(b)	450	2	30	0.0000	0.0000	0.0000	4:2:0	18
00228(P)	456	2	28	0.0000	0.0000	0.0000	4:2:0	16
00227(b)	454	3	30	0.0000	0.0000	0.0000	4:2:0	11
00230(P)	460	3	28	0.0000	0.0000	0.0000	4:2:0	13
00229(b)	458	4	30	0.0000	0.0000	0.0000	4:2:0	16
00232(P)	464	4	28	0.0000	0.0000	0.0000	4:2:0	16
00231(b)	462	5	30	0.0000	0.0000	0.0000	4:2:0	11
00234(P)	468	5	28	0.0000	0.0000	0.0000	4:2:0	13
00233(b)	466	6	30	0.0000	0.0000	0.0000	4:2:0	17
00236(P)	472	6	28	0.0000	0.0000	0.0000	4:2:0	15
00235(b)	470	7	30	0.0000	0.0000	0.0000	4:2:0	12
00238(P)	476	7	28	0.0000	0.0000	0.0000	4:2:0	13
00237(b)	474	8	30	0.0000	0.0000	0.0000	4:2:0	17
00240(P)	480	8	28	0.0000	0.0000	0.0000	4:2:0	16
00239(b)	478	9	30	0.0000	0.0000	0.0000	4:2:0	11
00242(P)	484	9	28	0.0000	0.0000	0.0000	4:2:0	13
00241(b)	482	10	30	0.0000	0.0000	0.0000	4:2:0	16
00244(P)	488	10	28	0.0000	0.0000	0.0000	4:2:0	16
00243(b)	486	11	30	0.0000	0.0000	0.0000	4:2:0	11
00246(P)	492	11	28	0.0000	0.0000	0.0000	4:2:0	13
00245(b)	490	12	30	0.0000	0.0000	0.0000	4:2:0	17
00248(P)	496	12	28	0.0000	0.0000	0.0000	4:2:0	15
00247(b)	494	13	30	0.0000	0.0000	0.0000	4:2:0	11
00250(P)	500	13	28	0.0000	0.0000	0.0000	4:2:0	13
00249(b)	498	14	30	0.0000	0.0000	0.0000	4:2:0	17
00252(P)	504	14	28	0.0000	0.0000	0.0000	4:2:0	15
00251(b)	502	15	30	0.0000	0.0000	0.0000	4:2:0	12
00254(P)	508	15	28	0.0000	0.0000	0.0000	4:2:0	13
00253(b)	506	0	30	0.0000	0.0000	0.0000	4:2:0	16
00256(P)	512	0	28	0.0000	0.0000	0.0000	4:2:0	15
00255(b)	510	1	30	0.0000	0.0000	0.0000	4:2:0	12
00258(P)	516	1	28	0.0000	0.0000	0.0000	4:2:0	13
00257(b)	514	2	30	0.0000	0.0000	0.0000	4:2:0	16
00260(P)	520	2	28	0.0000	0.0000	0.0000	4:2:0	15
00259(b)	518	3	30	0.0000	0.0000	0.0000	4:2:0	11
00262(P)	524	3	28	0.0000	0.0000	0.0000	4:2:0	13
00261(b)	522	4	30	0.0000	0.0000	0.0000	4:2:0	17
00264(P)	528	4	28	0.0000	0.0000	0.0000	4:2:0	15
00263(b)	526	5	30	0.0000	0.0000	0.0000	4:2:0	12
00266(P)	532	5	28	0.0000	0.0000	0.0000	4:2:0	14
00265(b)	530	6	30	0.0000	0.0000	0.0000	4:2:0	17
00268(P)	536	6	28	0.0000	0.0000	0.0000	4:2:0	14
00267(b)	534	7	30	0.0000	0.0000	0.0000	4:2:0	11
00270(P)	540	7	28	0.0000	0.0000	0.0000	4:2:0	14
00269(b)	538	8	30	0.0000	0.0000	0.0000	4:2:0	17
00272(P)	544	8	28	0.0000	0.0000	0.0000	4:2:0	15
00271(b)	542	9	30	0.0000	0.0000	0.0000	4:2:0	12
00274(P)	548	9	28	0.0000	0.0000	0.0000	4:2:0	13
00273(b)	546	10	30	0.0000	0.0000	0.0000	4:2:0	17
00276(P)	552	10	28	0.0000	0.0000	0.0000	4:2:0	14

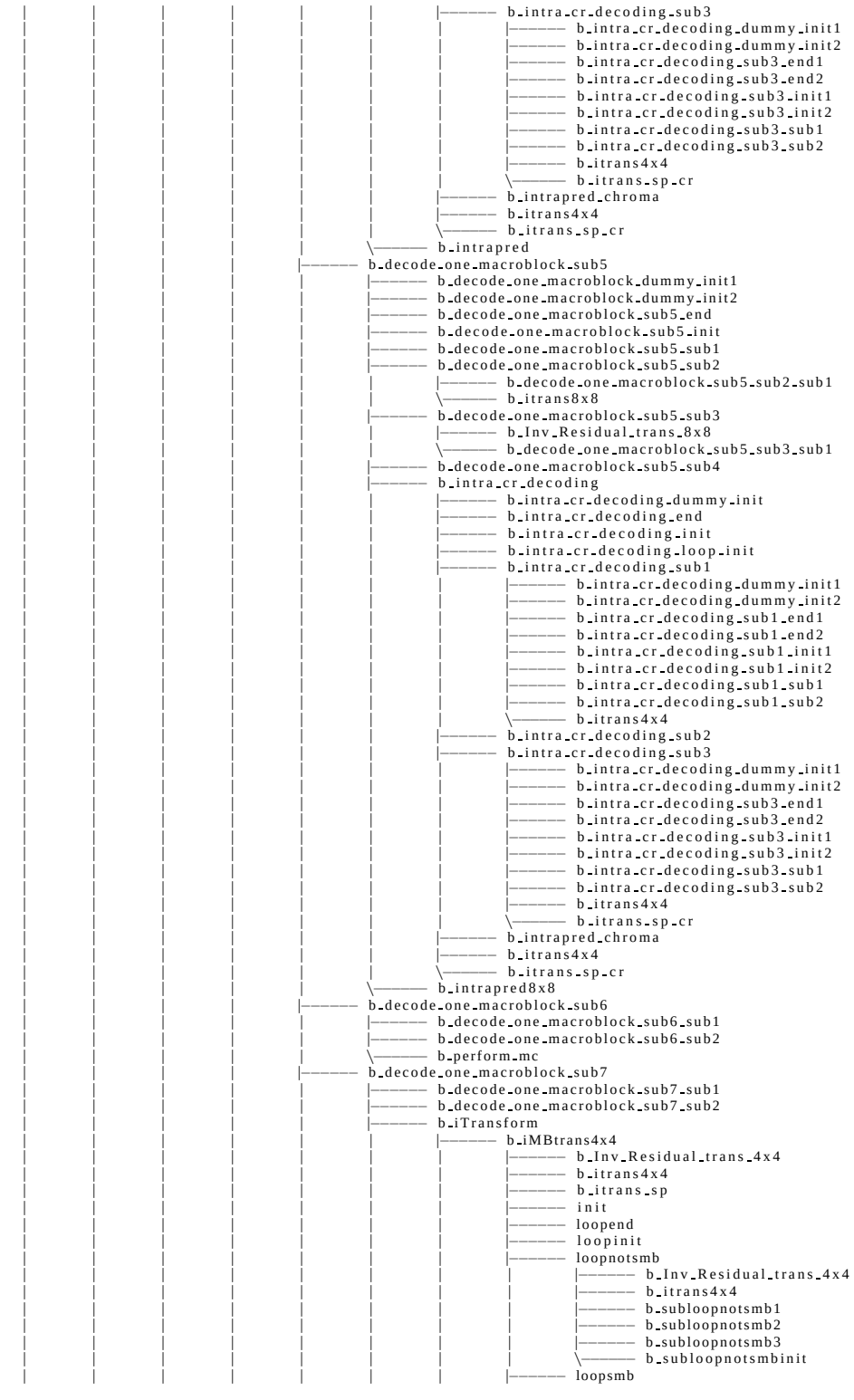
```
00275( b )       550    11     30    0.0000    0.0000    0.0000   4:2:0       12
00278( P )       556    11     28    0.0000    0.0000    0.0000   4:2:0       13
00277( b )       554    12     30    0.0000    0.0000    0.0000   4:2:0       17
00280( P )       560    12     28    0.0000    0.0000    0.0000   4:2:0       13
00279( b )       558    13     30    0.0000    0.0000    0.0000   4:2:0       12
00282( P )       564    13     28    0.0000    0.0000    0.0000   4:2:0       13
00281( b )       562    14     30    0.0000    0.0000    0.0000   4:2:0       16
00284( P )       568    14     28    0.0000    0.0000    0.0000   4:2:0       14
00283( b )       566    15     30    0.0000    0.0000    0.0000   4:2:0       12
00286( P )       572    15     28    0.0000    0.0000    0.0000   4:2:0       12
00285( b )       570     0     30    0.0000    0.0000    0.0000   4:2:0       17
00288( P )       576     0     28    0.0000    0.0000    0.0000   4:2:0       13
00287( b )       574     1     30    0.0000    0.0000    0.0000   4:2:0       12
00290( P )       580     1     28    0.0000    0.0000    0.0000   4:2:0       13
00289( b )       578     2     30    0.0000    0.0000    0.0000   4:2:0       16
00292( P )       584     2     28    0.0000    0.0000    0.0000   4:2:0       15
00291( b )       582     3     30    0.0000    0.0000    0.0000   4:2:0       11
00294( P )       588     3     28    0.0000    0.0000    0.0000   4:2:0       13
00293( b )       586     4     30    0.0000    0.0000    0.0000   4:2:0       16
00296( P )       592     4     28    0.0000    0.0000    0.0000   4:2:0       15
00295( b )       590     5     30    0.0000    0.0000    0.0000   4:2:0       11
00298( P )       596     5     28    0.0000    0.0000    0.0000   4:2:0       12
end decoding
---------------------- Average SNR all frames -----------------------------------
 SNR Y(dB)             :  0.00
 SNR U(dB)             :  0.00
 SNR V(dB)             :  0.00
 Total decoding time : 4.096 sec (72.998 fps)
---------------------------------------------------------------------------------
 Exit JM 13 (FREXt) decoder, ver 13.0
```

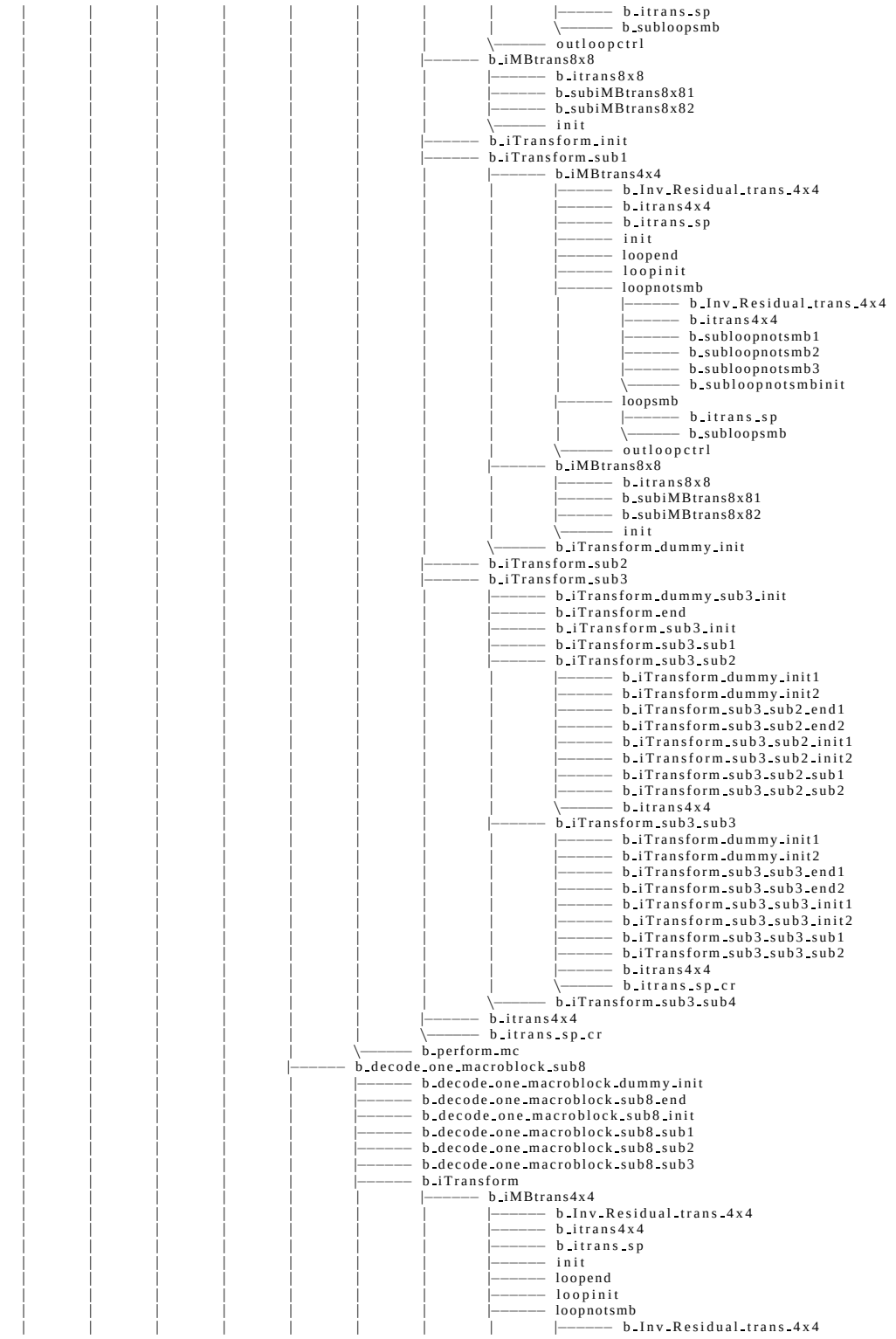
A.5 Clean behavioral hierarchy tree

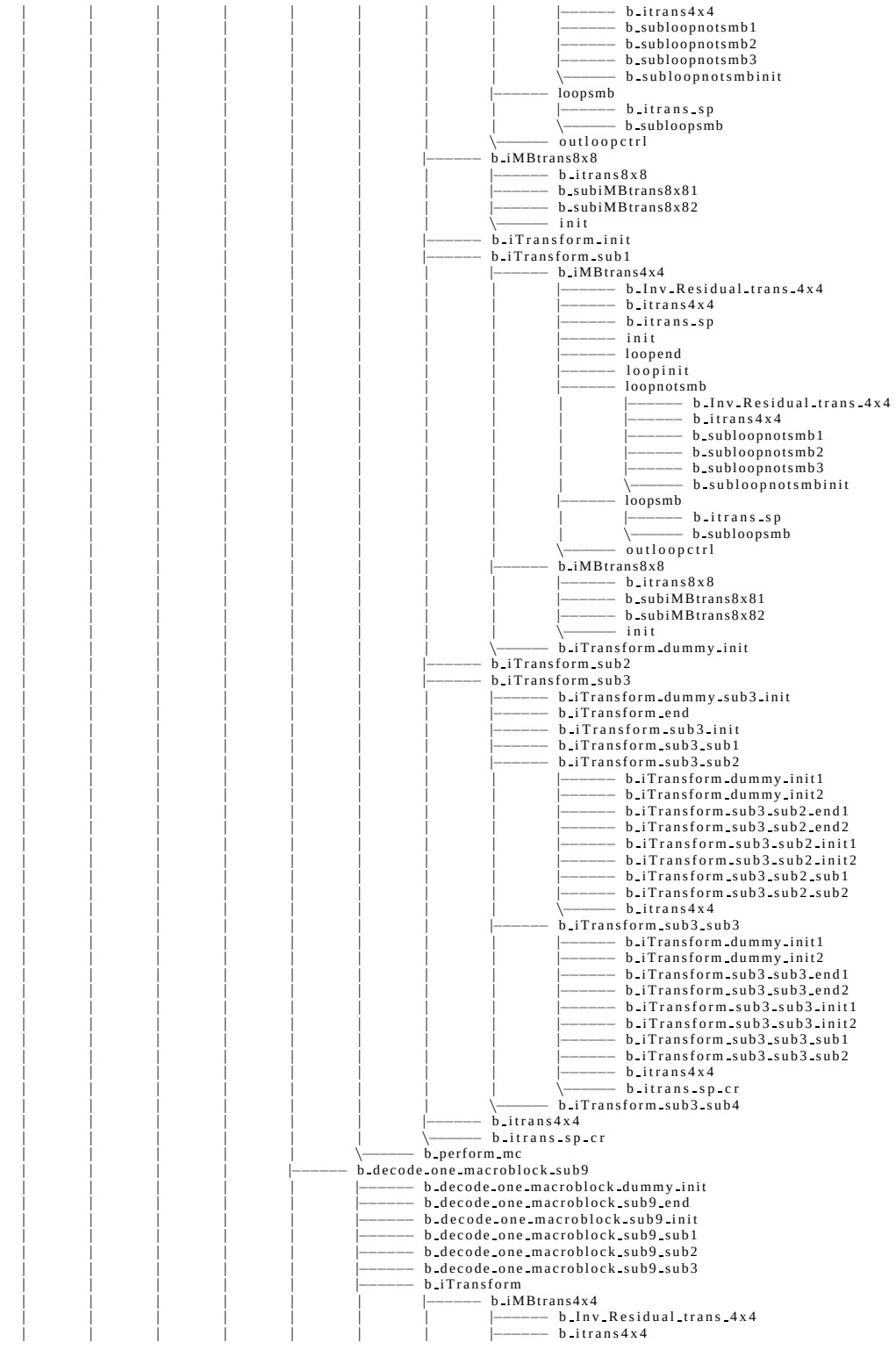
```
Main
|------ h264dec_DUT
|       |------ b_decode_slice
|       |       |------ b_decode_one_slice
|       |       |       |------ b_change_plane_JV
|       |       |       |------ b_compute_colocated
|       |       |       |------ b_compute_colocated_JV
|       |       |       |------ b_decode_one_macroblock
|       |       |       |       |------ b_Inv_Residual_trans_4x4
|       |       |       |       |------ b_Inv_Residual_trans_8x8
|       |       |       |       |------ b_decode_ipcm_mb
|       |       |       |       |------ b_decode_one_macroblock_sub1
|       |       |       |       |------ b_decode_one_macroblock_sub10
|       |       |       |       |       |------ b_decode_one_macroblock_dummy_init1
|       |       |       |       |       |------ b_decode_one_macroblock_sub10_end1
|       |       |       |       |       |------ b_decode_one_macroblock_sub10_init1
|       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub1
|       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub2
|       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub3
|       |       |       |       |       |       |------ b_decode_one_macroblock_dummy_init
|       |       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub3_end
|       |       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub3_init
|       |       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub3_sub1
|       |       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub3_sub2
|       |       |       |       |       |       \------ b_perform_mc
|       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub4
|       |       |       |       |       |       |------ b_decode_one_macroblock_dummy_init
|       |       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub4_end
|       |       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub4_init
|       |       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub4_sub1
|       |       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub4_sub2
|       |       |       |       |       |       \------ b_perform_mc
|       |       |       |       |       |------ b_decode_one_macroblock_sub10_sub5
|       |       |       |       |       |------ b_iTransform
|       |       |       |       |       |       |------ b_iMBtrans4x4
|       |       |       |       |       |       |       |------ b_Inv_Residual_trans_4x4
|       |       |       |       |       |       |       |------ b_itrans4x4
```

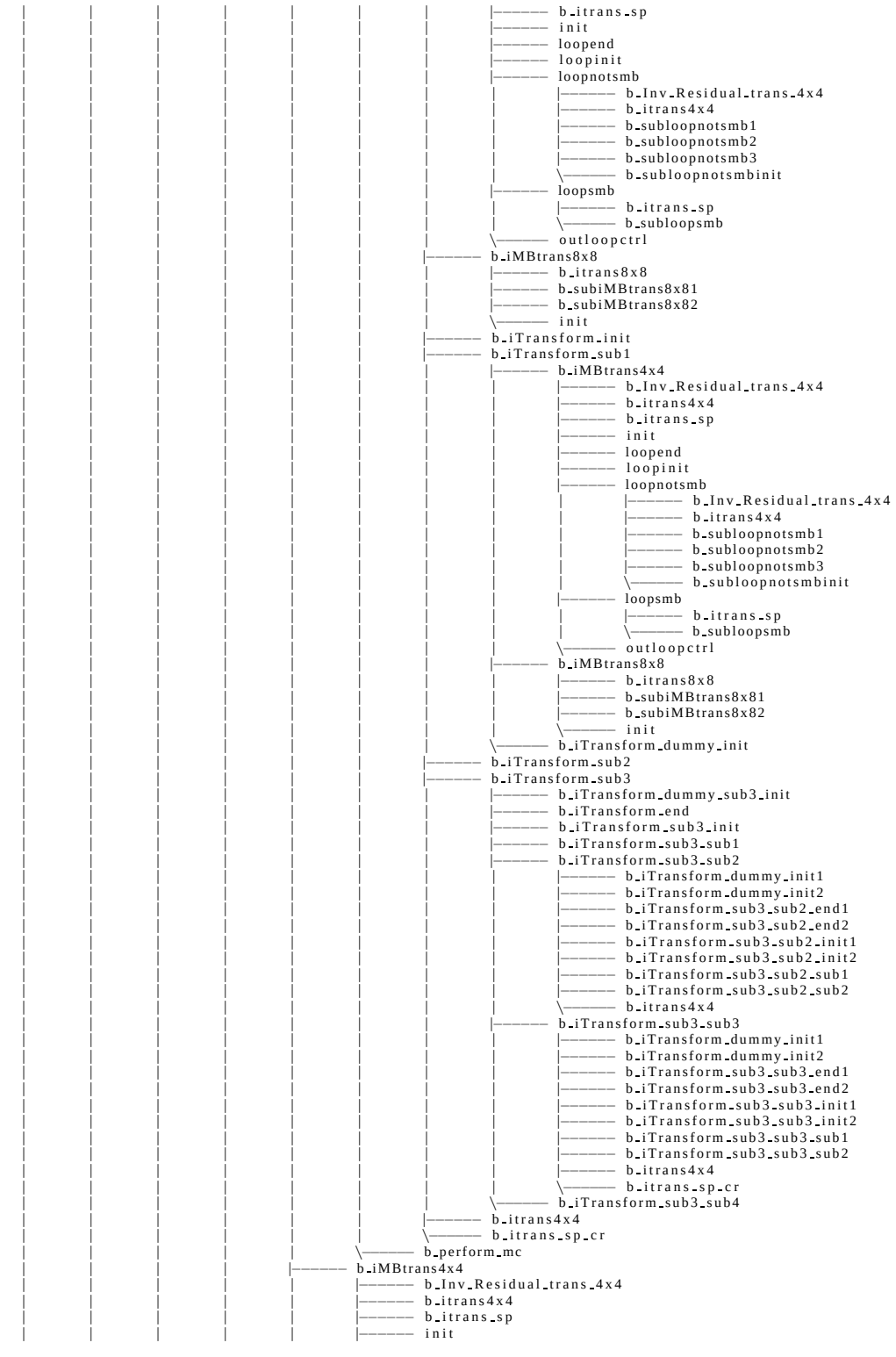

b_itrans_sp
init
loopend
loopinit
loopnotsmb
b_Inv_Residual_trans_4x4
b_itrans4x4
b_subloopnotsmb1
b_subloopnotsmb2
b_subloopnotsmb3
b_subloopnotsmbinit
loopsmb
b_itrans_sp
b_subloopsmb
outloopctrl
b_iMBtrans8x8
b_itrans8x8
b_subiMBtrans8x81
b_subiMBtrans8x82
init
b_iTransform_init
b_iTransform_sub1
b_iMBtrans4x4
b_Inv_Residual_trans_4x4
b_itrans4x4
b_itrans_sp
init
loopend
loopinit
loopnotsmb
b_Inv_Residual_trans_4x4
b_itrans4x4
b_subloopnotsmb1
b_subloopnotsmb2
b_subloopnotsmb3
b_subloopnotsmbinit
loopsmb
b_itrans_sp
b_subloopsmb
outloopctrl
b_iMBtrans8x8
b_itrans8x8
b_subiMBtrans8x81
b_subiMBtrans8x82
init
b_iTransform_dummy_init
b_iTransform_sub2
b_iTransform_sub3
b_iTransform_dummy_sub3_init
b_iTransform_end
b_iTransform_sub3_init
b_iTransform_sub3_sub1
b_iTransform_sub3_sub2
b_iTransform_dummy_init1
b_iTransform_dummy_init2
b_iTransform_sub3_sub2_end1
b_iTransform_sub3_sub2_end2
b_iTransform_sub3_sub2_init1
b_iTransform_sub3_sub2_init2
b_iTransform_sub3_sub2_sub1
b_iTransform_sub3_sub2_sub2
b_itrans4x4
b_iTransform_sub3_sub3
b_iTransform_dummy_init1
b_iTransform_dummy_init2
b_iTransform_sub3_sub3_end1
b_iTransform_sub3_sub3_end2
b_iTransform_sub3_sub3_init1
b_iTransform_sub3_sub3_init2
b_iTransform_sub3_sub3_sub1
b_iTransform_sub3_sub3_sub2
b_itrans4x4
b_itrans_sp_cr
b_iTransform_sub3_sub4
b_itrans4x4
b_itrans_sp_cr
b_perform_mc
b_prepare_direct_params
b_decode_one_macroblock_sub2
b_decode_one_macroblock_sub3
b_decode_one_macroblock_sub3_sub1
b_decode_one_macroblock_sub3_sub2

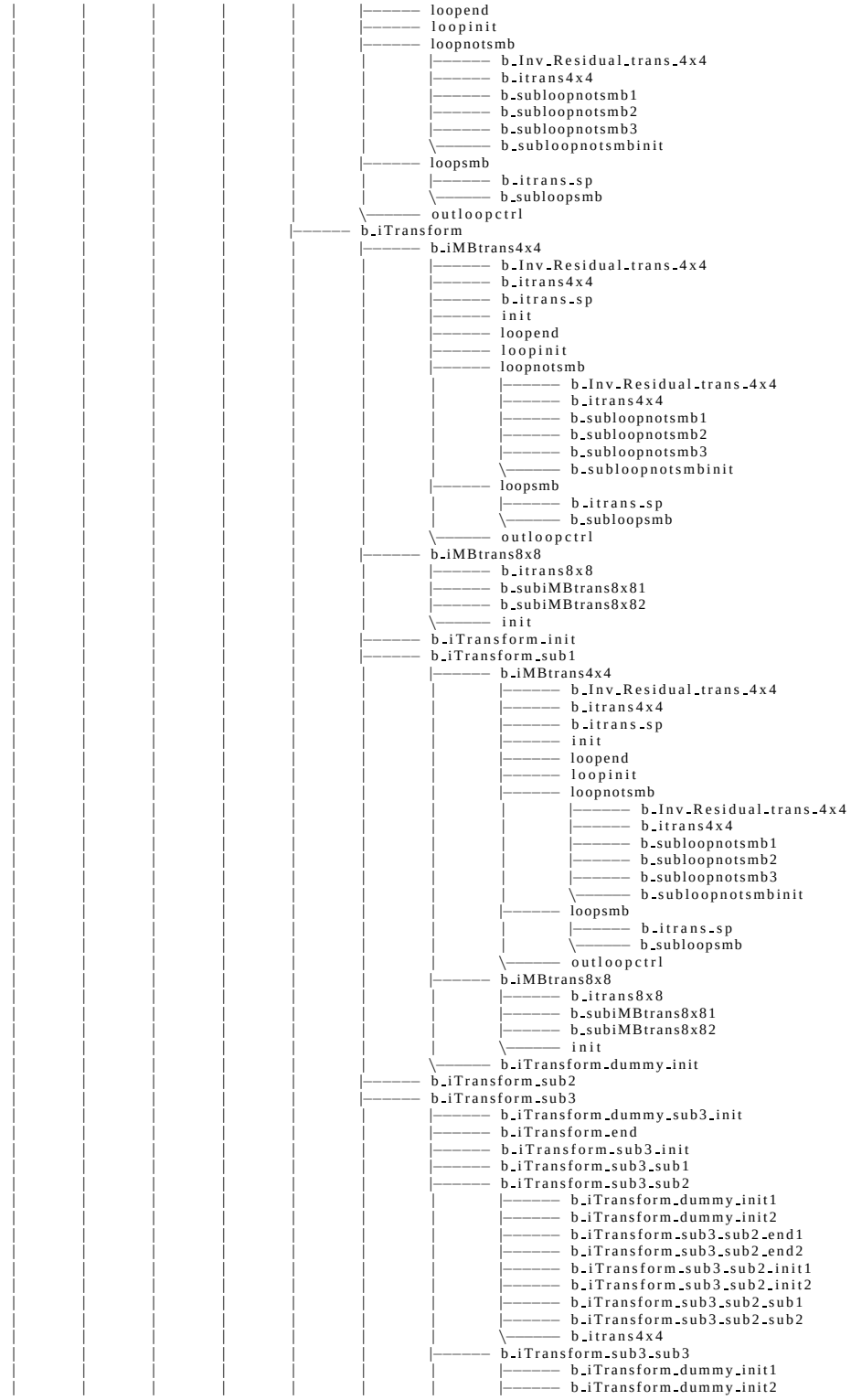

b_iMBtrans4x4
b_Inv_Residual_trans_4x4
b_itrans4x4
b_itrans_sp
init
loopend
loopinit
loopnotsmb
b_Inv_Residual_trans_4x4
b_itrans4x4
b_subloopnotsmb1
b_subloopnotsmb2
b_subloopnotsmb3
b_subloopnotsmbinit
loopsmb
b_itrans_sp
b_subloopsmb
outloopctrl
b_intra_cr_decoding
b_intra_cr_decoding_dummy_init
b_intra_cr_decoding_end
b_intra_cr_decoding_init
b_intra_cr_decoding_loop_init
b_intra_cr_decoding_sub1
b_intra_cr_decoding_dummy_init1
b_intra_cr_decoding_dummy_init2
b_intra_cr_decoding_sub1_end1
b_intra_cr_decoding_sub1_end2
b_intra_cr_decoding_sub1_init1
b_intra_cr_decoding_sub1_init2
b_intra_cr_decoding_sub1_sub1
b_intra_cr_decoding_sub1_sub2
b_itrans4x4
b_intra_cr_decoding_sub2
b_intra_cr_decoding_sub3
b_intra_cr_decoding_dummy_init1
b_intra_cr_decoding_dummy_init2
b_intra_cr_decoding_sub3_end1
b_intra_cr_decoding_sub3_end2
b_intra_cr_decoding_sub3_init1
b_intra_cr_decoding_sub3_init2
b_intra_cr_decoding_sub3_sub1
b_intra_cr_decoding_sub3_sub2
b_itrans4x4
b_itrans_sp_cr
b_intrapred_chroma
b_itrans4x4
b_itrans_sp_cr
b_intrapred_luma_16x16
b_decode_one_macroblock_sub4
b_decode_one_macroblock_dummy_init1
b_decode_one_macroblock_dummy_init2
b_decode_one_macroblock_dummy_init3
b_decode_one_macroblock_sub4_end1
b_decode_one_macroblock_sub4_end2
b_decode_one_macroblock_sub4_init1
b_decode_one_macroblock_sub4_init2
b_decode_one_macroblock_sub4_sub1
b_decode_one_macroblock_sub4_sub2
b_decode_one_macroblock_sub4_sub2_sub1
b_decode_one_macroblock_sub4_sub2_sub2
b_itrans4x4
b_decode_one_macroblock_sub4_sub3
b_Inv_Residual_trans_4x4
b_decode_one_macroblock_sub4_sub3_sub1
b_decode_one_macroblock_sub4_sub4
b_intra_cr_decoding
b_intra_cr_decoding_dummy_init
b_intra_cr_decoding_end
b_intra_cr_decoding_init
b_intra_cr_decoding_loop_init
b_intra_cr_decoding_sub1
b_intra_cr_decoding_dummy_init1
b_intra_cr_decoding_dummy_init2
b_intra_cr_decoding_sub1_end1
b_intra_cr_decoding_sub1_end2
b_intra_cr_decoding_sub1_init1
b_intra_cr_decoding_sub1_init2
b_intra_cr_decoding_sub1_sub1
b_intra_cr_decoding_sub1_sub2
b_itrans4x4
b_intra_cr_decoding_sub2

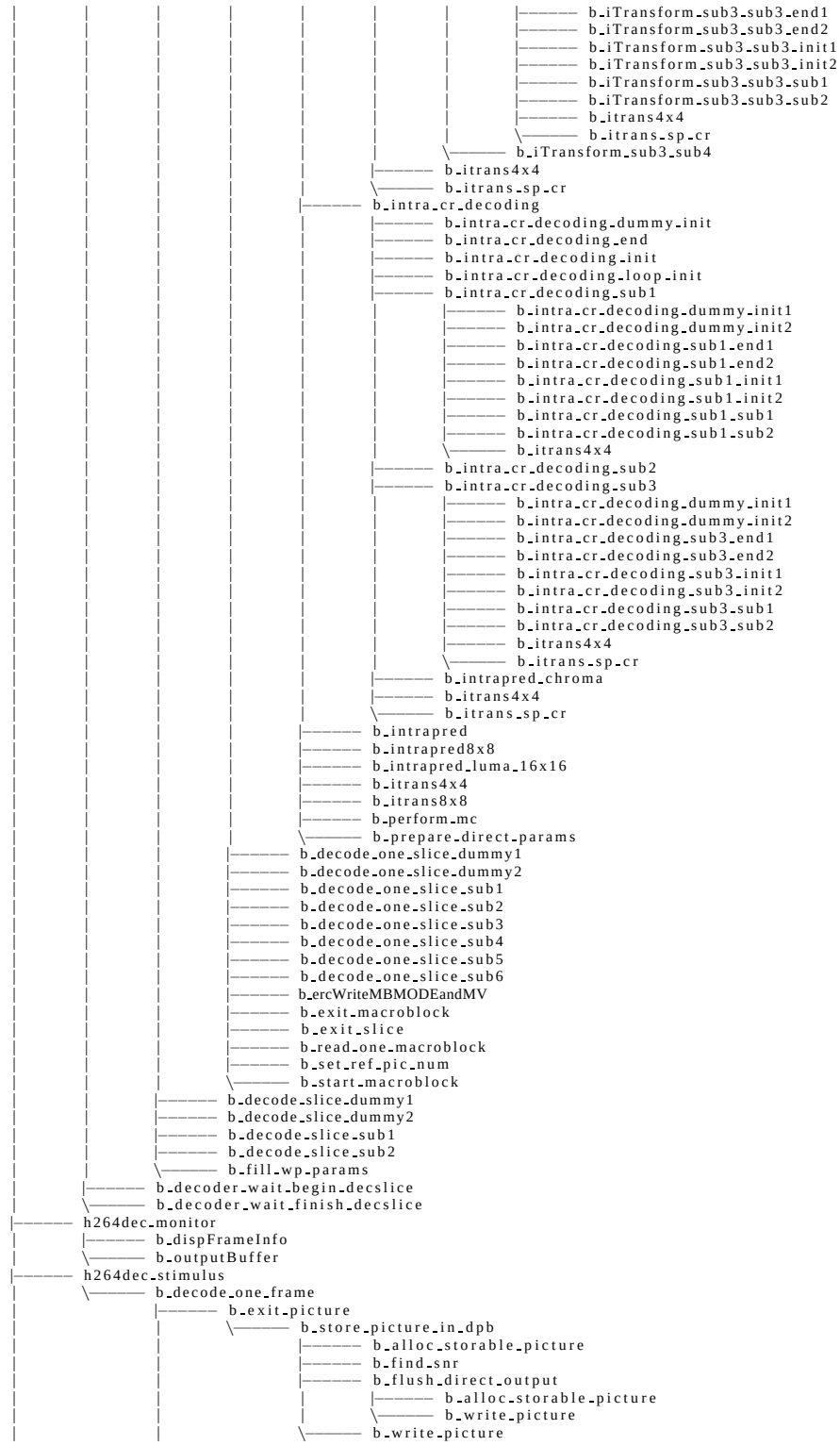

b_intra_cr_decoding_sub3
b_intra_cr_decoding_dummy_init1
b_intra_cr_decoding_dummy_init2
b_intra_cr_decoding_sub3_end1
b_intra_cr_decoding_sub3_end2
b_intra_cr_decoding_sub3_init1
b_intra_cr_decoding_sub3_init2
b_intra_cr_decoding_sub3_sub1
b_intra_cr_decoding_sub3_sub2
b_itrans4x4
b_itrans_sp_cr
b_intrapred_chroma
b_itrans4x4
b_itrans_sp_cr
b_intrapred
b_decode_one_macroblock_sub5
b_decode_one_macroblock_dummy_init1
b_decode_one_macroblock_dummy_init2
b_decode_one_macroblock_sub5_end
b_decode_one_macroblock_sub5_init
b_decode_one_macroblock_sub5_sub1
b_decode_one_macroblock_sub5_sub2
b_decode_one_macroblock_sub5_sub2_sub1
b_itrans8x8
b_decode_one_macroblock_sub5_sub3
b_Inv_Residual_trans_8x8
b_decode_one_macroblock_sub5_sub3_sub1
b_decode_one_macroblock_sub5_sub4
b_intra_cr_decoding
b_intra_cr_decoding_dummy_init
b_intra_cr_decoding_end
b_intra_cr_decoding_init
b_intra_cr_decoding_loop_init
b_intra_cr_decoding_sub1
b_intra_cr_decoding_dummy_init1
b_intra_cr_decoding_dummy_init2
b_intra_cr_decoding_sub1_end1
b_intra_cr_decoding_sub1_end2
b_intra_cr_decoding_sub1_init1
b_intra_cr_decoding_sub1_init2
b_intra_cr_decoding_sub1_sub1
b_intra_cr_decoding_sub1_sub2
b_itrans4x4
b_intra_cr_decoding_sub2
b_intra_cr_decoding_sub3
b_intra_cr_decoding_dummy_init1
b_intra_cr_decoding_dummy_init2
b_intra_cr_decoding_sub3_end1
b_intra_cr_decoding_sub3_end2
b_intra_cr_decoding_sub3_init1
b_intra_cr_decoding_sub3_init2
b_intra_cr_decoding_sub3_sub1
b_intra_cr_decoding_sub3_sub2
b_itrans4x4
b_itrans_sp_cr
b_intrapred_chroma
b_itrans4x4
b_itrans_sp_cr
b_intrapred8x8
b_decode_one_macroblock_sub6
b_decode_one_macroblock_sub6_sub1
b_decode_one_macroblock_sub6_sub2
b_perform_mc
b_decode_one_macroblock_sub7
b_decode_one_macroblock_sub7_sub1
b_decode_one_macroblock_sub7_sub2
b_iTransform
b_iMBtrans4x4
b_Inv_Residual_trans_4x4
b_itrans4x4
b_itrans_sp
init
loopend
loopinit
loopnotsmb
b_Inv_Residual_trans_4x4
b_itrans4x4
b_subloopnotsmb1
b_subloopnotsmb2
b_subloopnotsmb3
b_subloopnotsmbinit
loopsmb

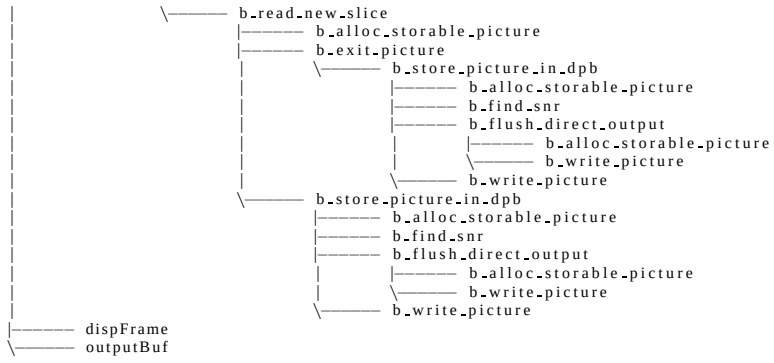

b_itrans_sp
b_subloopsmb
outloopctrl
b_iMBtrans8x8
b_itrans8x8
b_subiMBtrans8x81
b_subiMBtrans8x82
init
b_iTransform_init
b_iTransform_sub1
b_iMBtrans4x4
b_Inv_Residual_trans_4x4
b_itrans4x4
b_itrans_sp
init
loopend
loopinit
loopnotsmb
b_Inv_Residual_trans_4x4
b_itrans4x4
b_subloopnotsmb1
b_subloopnotsmb2
b_subloopnotsmb3
b_subloopnotsmbinit
loopsmb
b_itrans_sp
b_subloopsmb
outloopctrl
b_iMBtrans8x8
b_itrans8x8
b_subiMBtrans8x81
b_subiMBtrans8x82
init
b_iTransform_dummy_init
b_iTransform_sub2
b_iTransform_sub3
b_iTransform_dummy_sub3_init
b_iTransform_end
b_iTransform_sub3_init
b_iTransform_sub3_sub1
b_iTransform_sub3_sub2
b_iTransform_dummy_init1
b_iTransform_dummy_init2
b_iTransform_sub3_sub2_end1
b_iTransform_sub3_sub2_end2
b_iTransform_sub3_sub2_init1
b_iTransform_sub3_sub2_init2
b_iTransform_sub3_sub2_sub1
b_iTransform_sub3_sub2_sub2
b_itrans4x4
b_iTransform_sub3_sub3
b_iTransform_dummy_init1
b_iTransform_dummy_init2
b_iTransform_sub3_sub3_end1
b_iTransform_sub3_sub3_end2
b_iTransform_sub3_sub3_init1
b_iTransform_sub3_sub3_init2
b_iTransform_sub3_sub3_sub1
b_iTransform_sub3_sub3_sub2
b_itrans4x4
b_itrans_sp_cr
b_iTransform_sub3_sub4
b_itrans4x4
b_itrans_sp_cr
b_perform_mc
b_decode_one_macroblock_sub8
b_decode_one_macroblock_dummy_init
b_decode_one_macroblock_sub8_end
b_decode_one_macroblock_sub8_init
b_decode_one_macroblock_sub8_sub1
b_decode_one_macroblock_sub8_sub2
b_decode_one_macroblock_sub8_sub3
b_iTransform
b_iMBtrans4x4
b_Inv_Residual_trans_4x4
b_itrans4x4
b_itrans_sp
init
loopend
loopinit
loopnotsmb
b_Inv_Residual_trans_4x4


b_itrans4x4
b_subloopnotsmb1
b_subloopnotsmb2
b_subloopnotsmb3
b_subloopnotsmbinit
loopsmb
b_itrans_sp
b_subloopsmb
outloopctrl
b_iMBtrans8x8
b_itrans8x8
b_subiMBtrans8x81
b_subiMBtrans8x82
init
b_iTransform_init
b_iTransform_sub1
b_iMBtrans4x4
b_Inv_Residual_trans_4x4
b_itrans4x4
b_itrans_sp
init
loopend
loopinit
loopnotsmb
b_Inv_Residual_trans_4x4
b_itrans4x4
b_subloopnotsmb1
b_subloopnotsmb2
b_subloopnotsmb3
b_subloopnotsmbinit
loopsmb
b_itrans_sp
b_subloopsmb
outloopctrl
b_iMBtrans8x8
b_itrans8x8
b_subiMBtrans8x81
b_subiMBtrans8x82
init
b_iTransform_dummy_init
b_iTransform_sub2
b_iTransform_sub3
b_iTransform_dummy_sub3_init
b_iTransform_end
b_iTransform_sub3_init
b_iTransform_sub3_sub1
b_iTransform_sub3_sub2
b_iTransform_dummy_init1
b_iTransform_dummy_init2
b_iTransform_sub3_sub2_end1
b_iTransform_sub3_sub2_end2
b_iTransform_sub3_sub2_init1
b_iTransform_sub3_sub2_init2
b_iTransform_sub3_sub2_sub1
b_iTransform_sub3_sub2_sub2
b_itrans4x4
b_iTransform_sub3_sub3
b_iTransform_dummy_init1
b_iTransform_dummy_init2
b_iTransform_sub3_sub3_end1
b_iTransform_sub3_sub3_end2
b_iTransform_sub3_sub3_init1
b_iTransform_sub3_sub3_init2
b_iTransform_sub3_sub3_sub1
b_iTransform_sub3_sub3_sub2
b_itrans4x4
b_itrans_sp_cr
b_iTransform_sub3_sub4
b_itrans4x4
b_itrans_sp_cr
b_perform_mc
b_decode_one_macroblock_sub9
b_decode_one_macroblock_dummy_init
b_decode_one_macroblock_sub9_end
b_decode_one_macroblock_sub9_init
b_decode_one_macroblock_sub9_sub1
b_decode_one_macroblock_sub9_sub2
b_decode_one_macroblock_sub9_sub3
b_iTransform
b_iMBtrans4x4
b_Inv_Residual_trans_4x4
b_itrans4x4


b_itrans_sp
init
loopend
loopinit
loopnotsmb
b_Inv_Residual_trans_4x4
b_itrans4x4
b_subloopnotsmb1
b_subloopnotsmb2
b_subloopnotsmb3
b_subloopnotsmbinit
loopsmb
b_itrans_sp
b_subloopsmb
outloopctrl
b_iMBtrans8x8
b_itrans8x8
b_subiMBtrans8x81
b_subiMBtrans8x82
init
b_iTransform_init
b_iTransform_sub1
b_iMBtrans4x4
b_Inv_Residual_trans_4x4
b_itrans4x4
b_itrans_sp
init
loopend
loopinit
loopnotsmb
b_Inv_Residual_trans_4x4
b_itrans4x4
b_subloopnotsmb1
b_subloopnotsmb2
b_subloopnotsmb3
b_subloopnotsmbinit
loopsmb
b_itrans_sp
b_subloopsmb
outloopctrl
b_iMBtrans8x8
b_itrans8x8
b_subiMBtrans8x81
b_subiMBtrans8x82
init
b_iTransform_dummy_init
b_iTransform_sub2
b_iTransform_sub3
b_iTransform_dummy_sub3_init
b_iTransform_end
b_iTransform_sub3_init
b_iTransform_sub3_sub1
b_iTransform_sub3_sub2
b_iTransform_dummy_init1
b_iTransform_dummy_init2
b_iTransform_sub3_sub2_end1
b_iTransform_sub3_sub2_end2
b_iTransform_sub3_sub2_init1
b_iTransform_sub3_sub2_init2
b_iTransform_sub3_sub2_sub1
b_iTransform_sub3_sub2_sub2
b_itrans4x4
b_iTransform_sub3_sub3
b_iTransform_dummy_init1
b_iTransform_dummy_init2
b_iTransform_sub3_sub3_end1
b_iTransform_sub3_sub3_end2
b_iTransform_sub3_sub3_init1
b_iTransform_sub3_sub3_init2
b_iTransform_sub3_sub3_sub1
b_iTransform_sub3_sub3_sub2
b_itrans4x4
b_itrans_sp_cr
b_iTransform_sub3_sub4
b_itrans4x4
b_itrans_sp_cr
b_perform_mc
b_iMBtrans4x4
b_Inv_Residual_trans_4x4
b_itrans4x4
b_itrans_sp
init


loopend
loopinit
loopnotsmb
b_Inv_Residual_trans_4x4
b_itrans4x4
b_subloopnotsmb1
b_subloopnotsmb2
b_subloopnotsmb3
b_subloopnotsmbinit
loopsmb
b_itrans_sp
b_subloopsmb
outloopctrl
b_iTransform
b_iMBtrans4x4
b_Inv_Residual_trans_4x4
b_itrans4x4
b_itrans_sp
init
loopend
loopinit
loopnotsmb
b_Inv_Residual_trans_4x4
b_itrans4x4
b_subloopnotsmb1
b_subloopnotsmb2
b_subloopnotsmb3
b_subloopnotsmbinit
loopsmb
b_itrans_sp
b_subloopsmb
outloopctrl
b_iMBtrans8x8
b_itrans8x8
b_subiMBtrans8x81
b_subiMBtrans8x82
init
b_iTransform_init
b_iTransform_sub1
b_iMBtrans4x4
b_Inv_Residual_trans_4x4
b_itrans4x4
b_itrans_sp
init
loopend
loopinit
loopnotsmb
b_Inv_Residual_trans_4x4
b_itrans4x4
b_subloopnotsmb1
b_subloopnotsmb2
b_subloopnotsmb3
b_subloopnotsmbinit
loopsmb
b_itrans_sp
b_subloopsmb
outloopctrl
b_iMBtrans8x8
b_itrans8x8
b_subiMBtrans8x81
b_subiMBtrans8x82
init
b_iTransform_dummy_init
b_iTransform_sub2
b_iTransform_sub3
b_iTransform_dummy_sub3_init
b_iTransform_end
b_iTransform_sub3_init
b_iTransform_sub3_sub1
b_iTransform_sub3_sub2
b_iTransform_dummy_init1
b_iTransform_dummy_init2
b_iTransform_sub3_sub2_end1
b_iTransform_sub3_sub2_end2
b_iTransform_sub3_sub2_init1
b_iTransform_sub3_sub2_init2
b_iTransform_sub3_sub2_sub1
b_iTransform_sub3_sub2_sub2
b_itrans4x4
b_iTransform_sub3_sub3
b_iTransform_dummy_init1
b_iTransform_dummy_init2


b_iTransform_sub3_sub3_end1
b_iTransform_sub3_sub3_end2
b_iTransform_sub3_sub3_init1
b_iTransform_sub3_sub3_init2
b_iTransform_sub3_sub3_sub1
b_iTransform_sub3_sub3_sub2
b_itrans4x4
b_itrans_sp_cr
b_iTransform_sub3_sub4
b_itrans4x4
b_itrans_sp_cr
b_intra_cr_decoding
b_intra_cr_decoding_dummy_init
b_intra_cr_decoding_end
b_intra_cr_decoding_init
b_intra_cr_decoding_loop_init
b_intra_cr_decoding_sub1
b_intra_cr_decoding_dummy_init1
b_intra_cr_decoding_dummy_init2
b_intra_cr_decoding_sub1_end1
b_intra_cr_decoding_sub1_end2
b_intra_cr_decoding_sub1_init1
b_intra_cr_decoding_sub1_init2
b_intra_cr_decoding_sub1_sub1
b_intra_cr_decoding_sub1_sub2
b_itrans4x4
b_intra_cr_decoding_sub2
b_intra_cr_decoding_sub3
b_intra_cr_decoding_dummy_init1
b_intra_cr_decoding_dummy_init2
b_intra_cr_decoding_sub3_end1
b_intra_cr_decoding_sub3_end2
b_intra_cr_decoding_sub3_init1
b_intra_cr_decoding_sub3_init2
b_intra_cr_decoding_sub3_sub1
b_intra_cr_decoding_sub3_sub2
b_itrans4x4
b_itrans_sp_cr
b_intrapred_chroma
b_itrans4x4
b_itrans_sp_cr
b_intrapred
b_intrapred8x8
b_intrapred_luma_16x16
b_itrans4x4
b_itrans8x8
b_perform_mc
b_prepare_direct_params
b_decode_one_slice_dummy1
b_decode_one_slice_dummy2
b_decode_one_slice_sub1
b_decode_one_slice_sub2
b_decode_one_slice_sub3
b_decode_one_slice_sub4
b_decode_one_slice_sub5
b_decode_one_slice_sub6
b_ercWriteMBMODEandMV
b_exit_macroblock
b_exit_slice
b_read_one_macroblock
b_set_ref_pic_num
b_start_macroblock
b_decode_slice_dummy1
b_decode_slice_dummy2
b_decode_slice_sub1
b_decode_slice_sub2
b_fill_wp_params
b_decoder_wait_begin_decslice
b_decoder_wait_finish_decslice
h264dec_monitor
b_dispFrameInfo
b_outputBuffer
h264dec_stimulus
b_decode_one_frame
b_exit_picture
b_store_picture_in_dpb
b_alloc_storable_picture
b_find_snr
b_flush_direct_output
b_alloc_storable_picture
b_write_picture
b_write_picture


b_read_new_slice
b_alloc_storable_picture
b_exit_picture
b_store_picture_in_dpb
b_alloc_storable_picture
b_find_snr
b_flush_direct_output
b_alloc_storable_picture
b_write_picture
b_write_picture
b_store_picture_in_dpb
b_alloc_storable_picture
b_find_snr
b_flush_direct_output
b_alloc_storable_picture
b_write_picture
b_write_picture
dispFrame
outputBuf