INCREMENTAL HARDWARE DEVELOPMENT FROM MODULAR MIXED C-VHDL SIMULATION

¹Márlon A. Lorencetti, ²Wagston T. Staehler, ¹Altamiro A. Susin

¹Universidade Federal do Rio Grande do Sul
²Universidade Luterana do Brasil

ABSTRACT

Complex digital systems development needs large amounts of HDL code. Frequent compilations and simulations are time and CPU consuming activities. This paper presents a design flow that starts with a validated modular C-description and allows incremental HDL coding. The C-description is initially structured such that each module will become a hardware module, in a one to one correspondence, and then coded in C and validated using application specification or benchmark. One module at a time is coded in HDL, and validated by means of test vectors that are generated and analyzed by the system functions. These functions use the same variables to interface either the C or VHDL descriptions. The technique has been used to develop a H.264/AVC video decoder that uses thousands of test vectors.

1. INTRODUCTION

The presented method decreases the time used to descript and validate the hardware module, using a software model. While working as a model for the hardware description, the software is able to extract test vectors from the input data that will be used in the hardware validation.

This work was developed in the context of the Brazilian Digital Television System, the SBTVD (Sistema Brasileiro de Televisão Digital) [1]. The H.264/AVC (MPEG-4 part 10) standard [2] was chosen for the video coding in this new system for its high quality and compression rates, currently considered the "state-of-art" in video coding, achieving several improvements over previous standards. A consortium of brazilian universities is developing a video decoder that is compliant with this standard. Because software based versions of this decoder hardly achieve enough performance to work with high resolutions at real time, a hardware decoder [3] has been developed and prototyped in FPGA.

The decoding process is extremely modular, allowing different developers to implement different functionalities separetely, since the interface between neighbour modules is respected.

In order to help the development process, an equally modular software model was created and validated. Then, the software is used to generate input data and expected results to each module.

This paper presents the design flow used to implement a 4x4 intra frame prediction module in VHDL for a H.264 video decoder, using a previously developed model software [4].

2. H.264/AVC AND INTRA FRAME PREDICTION

The intent of the H.264/AVC project was to create a new standard capable of offering good video quality with lower bit rates than previous standards (MPEG-2, for example). Besides, the new standard had to provide flexibility to be used in different applications.

This standard is block-based, where an image is divided in blocks of 16x16 pixels, called macroblocks. The algorithms of encoding and decoding are applied separately for luma and chroma components, using the YCbCr color space. The data flow of this standard can be seen in the block diagram of the Fig. 1.

In order to compress the video data, H.264 brings some prediction methods to eliminate the redundancy that exists in the image data [6]. The temporal redundancy is explored with inter frame prediction methods, where a block is predicted as a similar blocks in previously decoded frames (from the past and/or future of the sequence) with an associated motion vector shifting its position. Another kind of prediction is the intra frame prediction, which uses the spatial similarities between neighbor blocks in the same frame. This kind of prediction is an innovation in video coding brought by H.264 standard.

Intra frame prediction is applied in 4x4, 8x8 or 16x16 pixels blocks with specific prediction directions. For the luma component of a macroblock the prediction can be executed in 4x4 blocks (nine modes available) or in the whole 16x16 block (four modes available). The 8x8 intra frame prediction is used in the sub-sampled chroma components of the macroblock and has four possible directions. The possible prediction directions for 4x4 luma blocks are shown in Fig. 2.

For the 4x4 intra prediction there is an algorithm to generate the expected prediction mode, because even the direction of intra prediction does not change so much in neighbor regions. To decide whether the predicted mode will be used or not, one bit is read from the bitstream. If the bit is '1', the predicted mode is used.

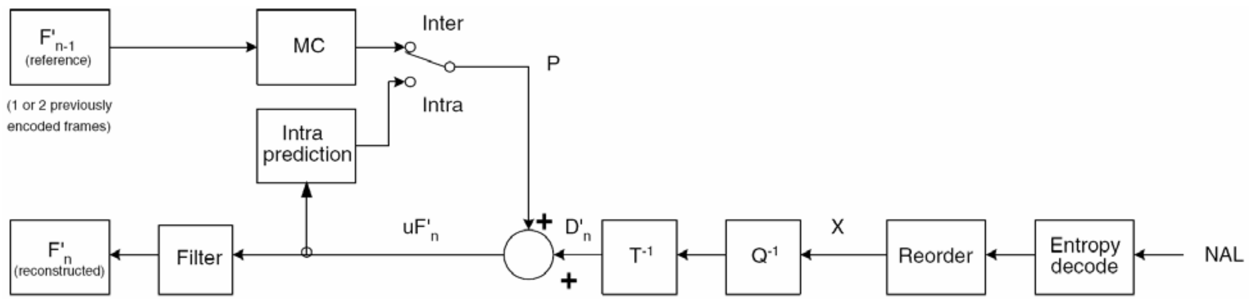

Fig. 1 – H.264/AVC block diagram.

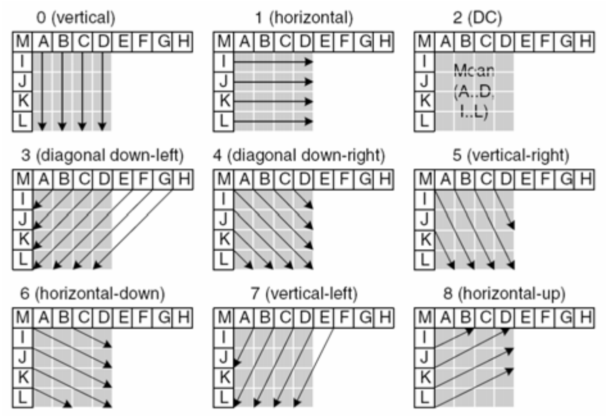

Fig. 2 - 4x4 Intra prediction modes.

Else, three additional bits are read to indicate the actual prediction mode.

3. SOFTWARE MODELING AND SIMULATION

The software model used in this work was already implemented in a previous work (more details in [4] and [5]). This software is used as a model instead of JM Reference Software [7] because the last one has a very difficult code, making its use as model software really tricky. The implemented software has a modular code that eased the extraction of sample data to support the hardware design.

Tab. 1 - Inputs and expected results.

Inputs	Expected Results
prediction bit: 1 Residual Luma 4x4 (128,160): -3 0 6 9 16 18 21 23 8 6 3 1 -21 -24 -30 -33	predicted mode 4x4: 1 mode 4x4: 1 Predicted Luma 4x4 (128,160): 106 106 106 106 60 60 60 60 92 92 92 92 140 140 140 140
prediction bit: 0 3 bits: 5 Residual Luma 4x4 (256,240): 18 7 1 6 5 1 7 19 -1 -3 11 25 6 1 7 18	predicted mode 4x4: 1 mode 4x4: 6 Predicted Luma 4x4 (256,240): 145 134 123 120 166 155 145 134 158 162 166 155 159 158 158 162

This software is capable of decoding bit streams coded with JM Reference Software containing only frames with intra prediction and CAVLC entropy coding. The implemented blocks are: CAVLC entropy decoder, intra prediction, inverse quantization, inverse transform and deblocking filter.

In order to have input data to test the hardware, some parameters are extracted from the software. For the example given on this paper (4x4 intra frame prediction) these parameters are: the bit indicating the use of 4x4 predicted intra mode, the other three bits to generate the new 4x4 prediction mode (when applicable), and all the residual data needed to reconstruct the pixels used to predict the next blocks.

Also, the expected results are generated in software. These results are: the predicted mode for 4x4 blocks, the used mode for 4x4 blocks, and the prediction results. An example of input data and expected results is shown on Tab. 1.

4. DESIGN METHODOLOGY

The H.264 decoder has several operating blocks, so the hardware development group is likely to adopt an incremental synthesis process. The problem about implementing the whole standard at once becomes even bigger when the matter is to debug the entire hardware in a HDL level.

In order to help the development of a hardware description a structured software was used, containing well defined interfaces linking the different modules. This software works as a reference model due to its more simple coding style, providing a better understanding of the algorithms used in the decoder. This software was implemented using the equations and descriptions of the set of recommendations of ITU-T for H.264 video decoder. This implementation was validated through output comparison with JM Reference Software, which is fully compliant with those recommendations.

After the software implementation, is possible to write a specification for the hardware module, because the software shows clearly the required variables and condition tests needed to implement the circuit. Besides, the creation of the software includes the creation of proposed test situations, which will be used to validate the hardware. The input data stream is extracted from the model software and stored into a file to be used to test the hardware. Also, the software generates the expected results for the corresponding input stream for that block.

Then, when the hardware implementation is done, the stored input data is adapted to fit the hardware input constraints and the block is simulated in the ModelSim Xilinx environment. The output of this simulation is stored and compared with its correspondent software generated expected results. After a set of tests containing images with different scenes, the hardware behavior is considered validated.

5. HARDWARE IMPLEMENTATION RESULTS

This work will use the implementation of a 4x4 intra predictor for a video decoder as an example of the presented methodology.

5.1. Luma 4x4 Intra Prediction – Architecture

An initial idea to implement this module is to create a different circuit to each prediction mode. This way, all the possible predictions are executed, but just the selected one passes to the output through a multiplexer.

Besides, the circuit could be completely parallelized, so a full 4x4 block prediction would take just one clock cycle to be completed. However, this implementation occupies a large area of the FPGA, leading to an ineffective way to perform the task, despite the high performance achieved.

Taking a better look at the equations used in the model software, or even at the ITU-T recommendations, is noticeable that for all intra prediction modes (except for the DC mode) an equation in the format of the equation (1) is used for all samples. So, the proposed implementation architecture explores this fact defining a structure to calculate this linear combination in a hardware block [8]. For the implementation of luma 4x4 prediction the proposed architecture uses a main processing element (PE) to implement the equation (1):

$$P_i = (n1 + c1*n2 + n3 + c2) >> c3 \quad (1)$$

The *n* values are neighbor pixels to this block, and the *c* values are constants. Depending on the chosen mode the values of *n1, n2, n3, c1, c2* and *c3* are switched. This equation is used to calculate every available mode, except the DC one. The operational part of the proposed architecture is shown in Fig. 3.

The DC mode is the arithmetic mean of the available neighbors. Besides, the DC value must be calculated in just one clock cycle. To do so, the architecture uses two PE units, selecting the proper constants when the DC mode is chosen. Then a multiplexer selects between the mean of the outputs of those PEs, the output of just one of them (if the block does not have all the neighbors),

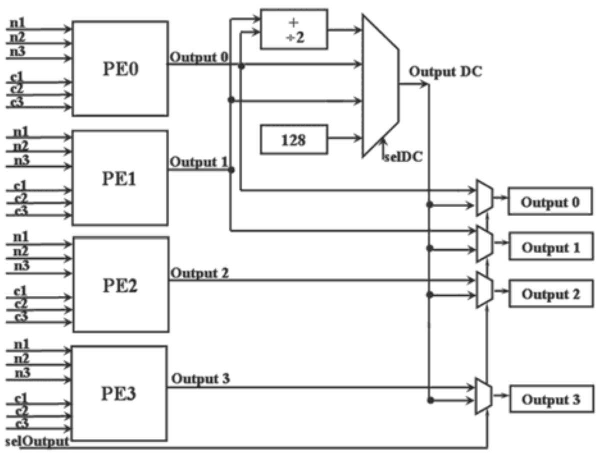

Fig. 3 - 4x4 Luma Prediction Architecture.

or the default 128 value (if there is no neighbors at all). In order to reach the needed throughput to be able to deal with high definition resolutions for television, the system has to generate 93.3 millions of samples per second (1920*1080 pixels *30 frames per second *1.5 luma and chroma samples). The proposed architecture processes four samples per cycle (using four PEs), so it can achieve HDTV rates running at 23.328MHz. This partially parallel implementation provides a balance of performance and occupied area of the FPGA [7].

5.2. Luma 4x4 Intra Prediction – Behavioral Validation

The input data set extracted from the software is adapted and introduced in the ModelSim VHDL simulation tool. The output signals of the simulation are captured into a file. Then, a simple software application compares theses signals with the expected results extracted from the software. In this implementation all the signals matched the expected software results in the many tests performed. Thus, the VHDL implementation of the intra prediction is now considered validated.

5.3. Luma 4x4 Intra Prediction – Synthesis Results

The hardware development group prototypes its modules in FPGA. The board used is a Digilent XUP V2P, with a Xilinx Virtex-II Pro XCV2VP30 FPGA. After synthesizing for this device, the module presented the results shown on Tab. 2:

Tab. 2 - Synthesis results.

FPGA Area (LUTs)	2323 of 27392 (8.48%)
FPGA Area (Flip-Flops)	219 of 27392 (0.80%)
Clock	336 MHz
Throughput	4 samples/cycle

As discussed before, the circuit had to work at 23.328MHz to reach the needed throughput for HDTV, but based on the synthesis results the module operates with a large time margin, and will not limit the speed of a complete decoder system.

6. CONCLUSIONS

This paper presented the design flow used to implement a 4x4 intra predictor module for an H.264 video decoder. A software version of the decoder was used as a reference model for the hardware development, providing a better understanding of the interfaces and methods used to predict the current block. Before the hardware implementation, this software was used to create a specification for the hardware design, since it is more compact and intuitive than the JM Reference Software or even the set of ITU-T recommendations, which has not a much straightforward algorithm description. Besides, after the hardware implementation, the same software is able to provide data vectors to validate the circuit in a behavioral simulation. The outputs of both software and hardware were compared in order to detect errors or confirm the accuracy of the hardware module.

As future works, more features could be included in the software, like the functions required to decode inter predicted frames. Also, more functions to acquire test data should be included, in order to support the debug and validation for all H.264 decoder modules. Another possible use for the software model is the validation of the whole decoder system after integration, using basically the same test strategy.

7. REFERENCES

[1] "Televisão Digital Terrestre – Codificação de vídeo, áudio e multiplexação", *ABNT*, Rio de Janeiro-RJ, 2007.

[2] Video Coding Experts Group, "ITU-T Recommendation H.264 (03/05): Advanced video coding for generic audiovisual services", *International Telecommunication Union*, 2005.

[3] L.V. Agostini et al, "Design and FPGA Prototyping of a H.264/AVC Main Profile Decoder for HDTV", *IEEE International Workshop on Rapid System Prototyping*, Proceedings, Porto Alegre-RS, pp. 174-180, 2007.

[4] M.A. Lorencetti, W.T. Staehler and A.A. Susin, "Reference C Software H.264/AVC Decoder for Hardware Debug and Validation", *XXIII South Symposium on Microelectronics*, SBC, Bento Gonçalves-RS, pp 127-130, 2008.

[5] M. Fiedler, "Implementation of a basis H.264/AVC Decoder", Seminar Paper, 2004.

[6] I.E.G. Richardson, "H.264 and MPEG-4 Video Compression: Video Coding for the Next-generation Multimedia", *John Wiley and Sons*, England, 2003.

[7] "H.264 Reference Software", http://iphom.hhi.de/suehring/tml/, 2008.

[8] W.T. Staehler and A.A. Susin, "Real-Time 4x4 Intraframe Prediction Architecture for an H.264 Decoder", *VI ITS*, IEEE International Telecommunications Symposium, Fortaleza-CE, 2006.