Network-on-Chip Based Architecture of H.264 Video Decoder

Adam Łuczak, Paweł Garstecki, Olgierd Stankiewicz and Marta Stepniewska.
Chair of Multimedia Telecommunications and Microelectronics,
Poznan University of Technology,
Polanka 3, 60-965 Poznań, Poland,
e-mail: [aluczak, pgarstec]@multimedia.edu.pl

***Abstract*— In this paper we describe architecture for H.264/AVC video decoder. This architecture exploits NoC (Network-on-Chip) for data transport between decoder blocks and is optimized for efficient processing, simple data flow and management. Proposed solution enables flexible device structure configuration and supports testing and verification environments. The presented original architecture is general and can be adopted to develop any modern video and audio codec.**

I. Introduction

In the recent years new video compression standards were released. These methods like H.264/AVC [1,2] (or VC-1, AVS) allow achieving high compression ratio and preserve very good image quality at the same time what makes them very attractive for various applications. Today also hardware implementations of i.e. H.264/AVC are one of the most required.

In [3,4,5,6] many hardware implementations of H.264/AVC decoder blocks as well as the whole decoder structures[7,8] were proposed. These architectures are not only optimized for efficient data processing but also low power consumption, data flow management or memory access [9] are taken under consideration.

In this paper we describe architecture development for modern video decoder such as H.264/AVC. We show structure supporting main features of the AVC standard and we point out most important issues of the top-level architecture design.

The main goal of our work is to create structure of video decoder that is flexible and can be easily configured in several ways. A typical video decoder implementation has very compact/tight formatted structure. It is difficult to adapt such a construction to other implementations. Thus, an important problem is development of the top model of a video codec. Many dependencies between modules become determined when choosing a particular construction. It affects processing efficiency, codec complexity, memory bandwidth and other features.

The original architecture we present is based on network-on-chip (NoC) designed especially for that purpose. This network is used to connect all decoder blocks and in this way we can construct modular video decoder architecture. NoC enables creating virtual channels between blocks. It is important that such virtual channel can connect two or more functional blocks. This is a unique feature of our original NoC network implementation.

Moreover, the architecture we propose can be reconfigured depending on decoder purpose or supported profile or application. This structure enables high processing efficiency, simple decoder management, extensive data flow control and overall communication.

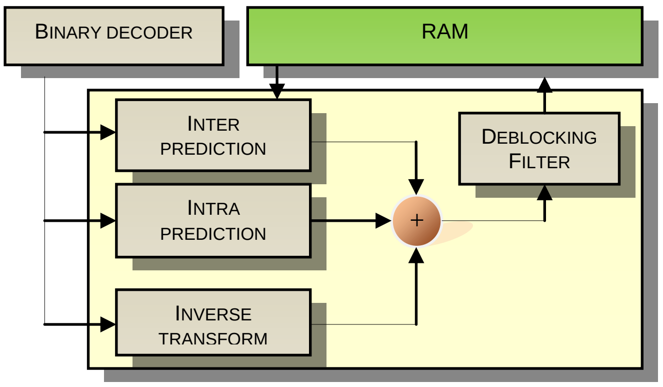

Fig. 1 Block diagram of H.264/AVC decoder

II. H.264/AVC compression issues

The H.264/AVC standard allows achieving very good compression ratio preserving high decoded picture quality at the same time. The good performance is possible because of many complex compression tools applied [10]. The standard defines sophisticated prediction methods, numerous macroblock coding modes, context dependent stream encoding and various binary coding schemes. Figure 1 shows a general block diagram of an AVC decoder. The presented blocks are: binary stream decoder, and reconstruction part including inverse transform, prediction blocks, and deblocking filter and RAM memory.

Decoder executes two main processes: binary stream decoding (parsing) and image reconstruction.

A. Binary stream decoding

Stream decoding is the first process performed and it is required to retrieve data from a binary stream used to decode and reconstruction frames of a video. As it is stream decoding procedure its complexity is determined by the stream syntax and coding tools (algorithm) used.

The H.264/AVC binary stream symbols are encoded in CAVLC (*context adaptive variable length coding*) or CABAC (*context adaptive binary arithmetic coding*) mode [2]. Depending on symbol type fixed length or variable length codes are used. According to the standard, the methods exploited are Exp-Golomb and Huffman or binary arithmetic coding and their variations.

Additionally encoding algorithm of many AVC syntax elements exploits prediction. This process is invoked for elements like motion vectors, transform coefficient count, or even intra prediction mode direction. Therefore, AVC stream decoding is highly context dependent. It means that in order to decode any value properly, we need to know values of previously decoded symbols. This implies storage of many parameters and high requirements for context memory capacity and data management.

Another issue is irregular AVC stream syntax that contains numerous elements that are present only under a particular condition and often just the previous syntax element determines this condition. This makes developing of architecture for efficient parsing process difficult.

B. Image reconstruction

The AVC defines numerous picture coding techniques that allow achieving high compression gain. In a decoder it must be possible to reconstruct a frame of a sequence that can be encoded in a scheme utilizing different modes (i.e. frame or field or adaptive frame/field) and prediction techniques (intra or inter). In the case of intra prediction, the decoder must support whole macroblock prediction mode (*Intra16x16*) and 4x4-prediction mode (*Intra4x4*) – each in several directions. In the case of inter prediction, it must allow using different macroblock partitioning and multiple reference frames. Decoder must also perform inverse coefficients transform. Basically only 4x4 inverse transform is used, however, depending on supported profile it may be necessary to implement an alternative 8x8 prediction and transform.

At the end of reconstruction procedure each frame may be filtered using a deblocking filter.

III. Architecture for Video Decoder

A block diagram of our proposed architecture and model of a testing environment is presented in figure 2. This structure was developed after a comprehensive analysis of stream parsing process and image reconstruction algorithm. Taking into consideration different character of both procedures, we proposed structure comprising two main blocks – parser and image samples reconstruction.

The main element of parser block is a microcontroller that controls the whole decoder and process data. It is a dedicated entity using a set of co-processor modules to perform specific operations like i.e. binary decoding – separate devices for CAVLC (*bdec_cavlc*) and CABAC (*bdec_cabac*), motion vector prediction (*mvp*) or NoC access (*noc_if*). Any of these devices is directly connected to microprocessor's device bus and NoC endpoint.

The image reconstruction part of the architecture contains modules performing samples predictions (*intra*, *inter*), an inverse transform (*itrans*), local context buffer (*pctx*), data merging unit (*merger*) and deblocking filter (*dblk*) as well as output buffer (*wcache*). All decoder modules are connected

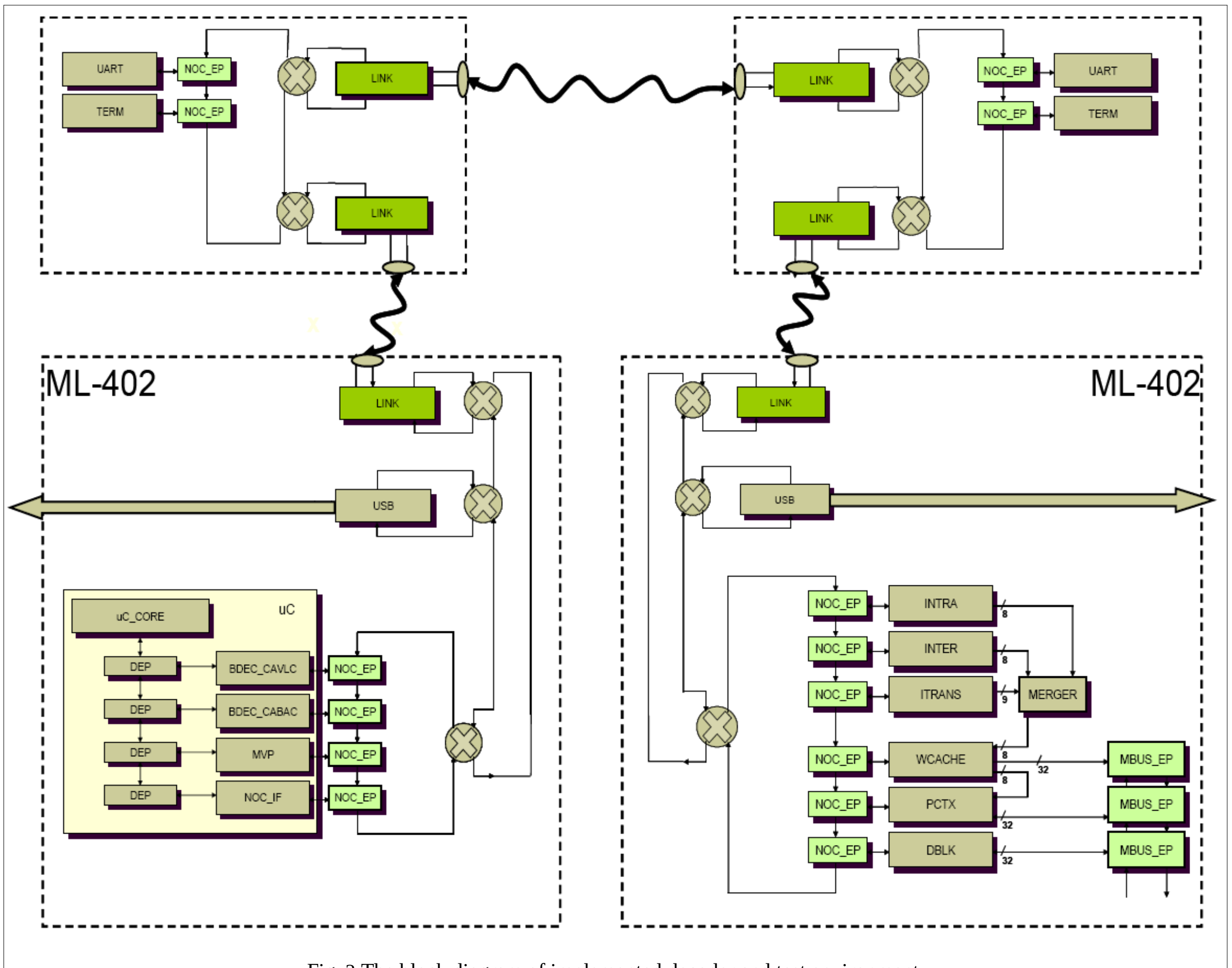

Fig. 2 The block diagram of implemented decoder and test environment.

using a dedicated network-on-chip (*NoC)* and special interface module (*noc_ep*) is used for that reason.

A. Flexible decoder configuration

The implemented NoC allows for flexible decoder (connections) configuration and adding/removing tools to/form the whole structure. Its most important feature is a structure that it is possible to separate local data transfers from global ones. This results in shorter transmission paths that can reduce delays, deadlocks and bandwidth in some areas of the decoder. Another important issue is flexibility that enables configuring new endpoints i.e. for debugging purpose or I/O operations

B. Efficient stream decoding

Stream parsing along with parameter prediction are performed in parallel. The main unit is a programmable microcontroller that runs decoding procedure. It performs binary stream decoding, syntax parsing and controls devices. In the parser module it drives binary decoders (*bdec_cavlc, bdec_cabac)* and parameters decoders (i.e. *mvp*). It also sends global parameters (i.e. frame size, frame slot and synchronization codes) to any reconstruction module.

Devices connected to the device bus support specific operations and make parser (microcontroller) capable of pipelined stream processing. Such construction increase processing efficiency significantly – especially in the case of elements implying prediction (i.e. motion vectors and intra prediction mode) or any other additional processing (i.e. deblocking filter strength) as it is a natural multiple stage process: i.e. decoding prediction error and computing final values. Processing results can be returned to the microprocessor unit (i.e. decoded syntax element value) or directly passed to the reconstruction blocks. It is possible because all devices are connected directly to NoC structure. The main idea of such a construction is avoiding unnecessary data transfers (through microcontroller) and flexible data processing. We avoid sequential processing loop in the decoder process which would be very probable scenario if all modules were managed from the main application level (μC code level).

C. Pipelined image reconstruction

Image reconstruction blocks process macroblocks independently and simultaneously. Processing starts if only all required input data or parameters for intra prediction or inter prediction or inverse transform are valid. Modules use small local FIFO queues for input parameters buffering and therefore processing can be carried out in all modules simultaneously. In order to keep proper macroblock processing order we use dedicated synchronization signals that carry information about current macroblock processing state. It means that macroblocks processing is pipelined and some macroblocks can be decoded in parallel – i.e. one inter and one intra. This results in good performance of the decoder. Reconstructed image samples are stored and ordered in a local memory buffer (*wcache).* It is used in order to maximize the efficiency of data writing procedure and utilize maximum abilities of RAM memory.

A part of reconstruction block is the deblocking filter procedure which is invoked after reconstruction of an image. This process may be considered detached therefore in our implementation deblocking filter is a separate entity. This module does not interact with other reconstruction blocks of the decoder except for checking data validity. Deblocking filtering is driven by the microcontroller which computes and sends filer input parameters (filter strength, thresholds). The module is capable of reading data from global memory as well as from local data buffer. In order to assure processing of valid frame data it utilizes a map of macroblock availability fags. In a consequence the filtering procedure is started as soon as possible and the delay is reduced.

D. Simple context management

As mentioned before stream parsing and image reconstruction require context data, however, each module uses a different set of parameters. In the case of stream parser, the required reference data are i.e. parameters describing encoding mode, motion vectors, prediction direction (from reference macroblocks or frames) while context used for image reconstruction comprises image samples. In our decoder image samples are stored in global RAM memory and context samples for intra prediction are buffered in a dedicated local buffer PCTX (*prediction context)* which supports signalling context data validity as well. This entity allows decreasing samples feedback latency. Parameters that built parser context are stored in microprocessor and global RAM memory in structures that enable simple context acquisition.

E. Memory configuration

Modules that require direct access to the global RAM memory are connected to a universal memory bus. This allows applying a standard interface in all entities and flexible data memory access management (i.e. setting priorities). The bus protocol is very simple and it can work with static and dynamic memories. It can be also easily adapted to connect with standard memory bus like AMBA.

F. Simple I/O interfacing

Connections between various modules are based on NoC and therefore it is easy to implement a direct input/output interface that may be used to set or get data from any particular entity. In the same way we may read or set data within any part of the global RAM memory or microcontroller memory (including μC program memory). These features are especially important when prototyping and debugging.

IV. Synthesis Results

We synthesized our decoder and implemented in on Xilinx Virtex 4 SX35 FPGA The synthesis results are presented in table 1.

TABLE I
Synthesis results for Virtex4 Sx 35 FPGA

Module	**Area**		
	4-input LUTs	**Registers**	**2K-Block RAMs**
Parser	~10000 (33%)	~6000 (20%)	69
Reconstruction	~10000 (32%)	~9000 (29%)	61 (31%)

Implemented video decoder can decode SDTV (standard definition television) binary streams at clock frequency about 66 MHz The clock frequency necessary to meet real-time SDTV processing requirements is so low that implemented decoder can be used also for mobile devices.

V. Conclusions

In this paper we presented architecture of AVC decoder based on network-on-chip. The main goal of our work was to create structure of video decoder that would be flexible and could be easily configured. We described structure of developed decoder and pointed main features of this design. We also implemented the proposed structure of the AVC decoder and tested it using FPGA devices (Xilinx Virtex4). It has been running successfully and its processing efficiency is high enough to effectively run decoding process of SDTV at very low clock frequency.

We proved that architecture introducing NoC network as main interconnection network allows building flexible and efficient video decoder. Moreover, it enables fast prototyping and easy and efficient debugging. The discussed structure is very general thus it may become a reference architecture for other video or even audio codec implementations.

References

[1] T. Wiegand, G. Sullivan, G. Bjøntegaard, and A. Luthra, Overview of the H.264/avc video coding standard. IEEE *Trans. on Circuits and Systems for Video Technology*, vol. 13, pp. 560–576, July 2003

[2] JVT of ISO/IEC MPEG & ITU-T VCEG *Text of ISO/IEC* 14496 10 Advanced Video Coding 3rd Edition, ISO/IEC JTC1/SC29/WG11, Redmond, July 2004.

[3] P. Garstecki and A. Łuczak, A flexible architecture for image reconstruction in h.264/avc decoders, in *Proc. of ECCTD 2005*, Cork, Ireland, August 2005.

[4] A. Łuczak, M. Stępniewska, ”Reconfigurable architecture for AVC/H.264 integer transform”, *European Signal Processing Conference,* pp. Sept. 2006

[5] Kun, Yang; Chun, Zhang; Zhihua, Wang, “Design of Adaptive Deblocking Filter for H.264/AVC Decoder SOC”, *IEEE Asia Pacific Conference on Circuits and Systems APCCAS,* pp.109 – 112, Dec 2006.

[6] Yi-Chih Chao; Shih-Tse Wei; Jar-Ferr Yang; Bin-Da Liu, “Combined Decoding and Flexible Transform Designs for Effective H.264/AVC Decoders”, *IEEE International Symposium on Circuits and Systems ISCAS 2007*, pp. 3135 – 3138, May 2007

[7] Ting-An Lin; Sheng-Zen Wang; Tsu-Ming Liu; Chen-Yi Lee, “An H.264/AVC decoder with 4/spl times/4-block level pipeline”, *IEEE International Symposium on Circuits and Systems ISCAS,* Vol. 2, pp. 1810 - 1813, May 2005

[8] To-Wei Chen; Yu-Wen Huang; Tung-Chien Chen; Yu-Han Chen; Chuan-Yung Tsai; Liang-Gee Chen, “Architecture design of H.264/AVC decoder with hybrid task pipelining for high definition videos”, *IEEE International Symposium on Circuits and Systems ISCAS*, Vol. 3, pp. 2931 – 2934, May 2005

[9] Shin-Haeng Ji; Jung-Wook Park; Shin-Dug Kim, “Optimization of Memory Management for H.264/AVC Decoder”, *The 8th International Conference on Advanced Communication Technology*, pp. 65-68, Feb. 2006

[10] M. Horovitz, A. Joch, F. Kossentini and A. Hallapuro, "H.264/AVC Baseline Profile Decoder Complexity Analysi*s", IEEE. Trans. on Circuits and Systems for Video Technology*, vol. 13, pp. 704_716, July 2003.