FPGA PROTOTYPING FOR FAST AND EFFICIENT VERIFICATION OF ASIC H.264 DECODER

by

BASAVARAJ MUDIGOUDAR

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2006

ACKNOWLEDGEMENTS

I am grateful to my advisor Dr. K. R. Rao for his continued encouragement and guidance throughout this thesis. Sincere thanks to UTA, my Alma Mater, for recognizing my aptitude and giving me an opportunity to pursue my master's studies under the guidance of noble faculty and for providing me with all the required resources and environment to excel.

I would also like to mention my gratitude to Dr. Pankaj Topiwala, president of FastVDO, for the support and trust he has shown in me to carry out this work and for all the resources provided at FastVDO facility. I am also grateful to Mr. Srinath Bagal and the entire ASIC team at FastVDO, without their cooperation this thesis would not be complete. Thanks to all my colleagues at FastVDO for all their help and cooperation in completion of this thesis.

Finally, I would like to take this opportunity to thank my family who has always supported me in my endeavors, without whom it would not have been possible to come this far.

April 17, 2006

ABSTRACT

FPGA PROTOTYPING FOR FAST AND EFFICIENT

VERIFICATION OF ASIC H.264 DECODER

Publication No. _____

Basavaraj Mudigoudar, MS

The University of Texas at Arlington, 2006

Supervising Professor: Dr. K. R. Rao

To improve compression efficiency, recent video compression standards such as H.264 use complex algorithms and various modes that demand more computational power. Consumer electronics industry requires a low power, compact and cost-effective implementation of video codec for most of the products. ASIC implementation of these video codecs is a logical choice to meet these requirements. Functional verification of an ASIC implementation consumes a major part of design cycle time and lot of resources. Because of large design, various modes and options, functional verification of an ASIC H.264 video codec is a challenging, resource intensive and time consuming

process. In this thesis an FPGA prototyping based functional verification technique has been suggested as fast and efficient alternative for functional verification of ASIC video codec. An FPGA prototyping of H.264 video codec has been performed for functional verification of ASIC video codec. Advantages and limitations have been elaborated with experimental results.

Keywords: H.264, AVC, MPEG-4 part 10, Functional verification, FPGA, ASIC Prototyping.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

ACRONYMS AND ABBREVIATIONS

ASIC: Application specific Integrated circuits

AVC: Advanced Video Coding

CABAC: Context-based Adaptive Binary Arithmetic Coding

CAVLC: Context-based Adaptive Variable Length Coding

DCT: Discrete Cosine Transform

DSP: Digital Signal Processors

DUT: Design Under Test

DVD: Digital Versatile Disc

EDA: Electronic Design Automation

FPGA: Field Programmable Gate Arrays

HD: High Definition

HDL: Hardware Description Language

IEC: International Engineering Consortium

ISO: International Standards Organization

ITU: International Telecommunication Union

JVT: Joint Video Team

MPEG: Moving Picture Experts Group

NALU: Network Abstraction Layer Unit

RTL: Register Transfer Logic

VCEG: Video Coding Experts Group

VHDL: VHSIC Hardware Description Language

VHSIC: Very High Speed Integrated Circuits

VLC: Variable Length Coding

VLSI: Very Large Scale Integration

CHAPTER 1

INTRODUCTION

Recent video compression standards such as MPEG-4 part 10 [1]/ H.264 [2] use advanced algorithms, various modes and profiles to give better compression efficiency. Adversely, these advanced techniques have increased the complexity of the compression process and demand more computational power. Conventional DSP based devices do not have enough processing power to handle the complexity of the algorithms at higher resolutions and the consumer electronics industry needs a low power, compact and cost-effective implementation of the video codec. ASIC implementation of the video codec is one of the logical choices to meet these requirements. It is an industry statistic that functional verification takes around 70% of ASIC design cycle time and requires lot of resources [3]. Because of large design, various modes and profiles, functional verification of an ASIC H.264 video codec is a challenging, resource intensive and time consuming process. In this thesis an FPGA prototyping based functional verification technique is suggested as a fast and efficient alternative for functional verification of ASIC video codec. An FPGA prototyping of H.264 video codec is performed and an extensive study is done on FPGA prototyping based functional verification of ASIC video codec. Advantages and disadvantages are elaborated with experimental results.

1.1 Thesis outline

Chapter 2 gives an overview of the H.264 standard and explains the ASIC implementation of the standard. Coding algorithms used in H.264 and ASIC design process are briefed.

Chapter 3 explores the ASIC verification techniques commonly used in the industry and discusses the advantages and disadvantages. Common methodologies and practices are introduced with emphasis to FPGA based prototyping.

In Chapter 4 FPGA prototyping of an ASIC video codec is presented. This chapter also discusses FPGA prototyping boards and their useful features. Methodologies used for verification of ASIC H.264 decoder are presented.

Chapter 5 outlines the results and advantages of FPGA prototyping based verification of an ASIC video codec. Results presented show the improvements achieved using the suggested techniques. Conclusions and future research interests are suggested.

CHAPTER 2

OVERVIEW OF H.264 AND ASIC IMPLEMENTATION

MPEG-4 part 10 or H.264 is the next generation video codec jointly developed by MPEG of ISO/IEC and VCEG of ITU-T [4]. The Joint Video Team (JVT) released the first draft of H.264, also known as Advanced Video Coding (AVC), in May 2003 [2] [5]. Juxtaposed to previous video compression standards such as MPEG-2 [6], MPEG-4 part 2 [7], H.264 also uses hybrid block based video compression techniques such as transformation for reduction of spatial correlation, quantization for bit-rate control, motion compensated prediction for reduction of temporal correlation and entropy coding for reduction in statistical correlation [4]. However, H.264 incorporates advanced algorithms and techniques to enhance coding performance over the previous standards and it gives better coding efficiency over MPEG-2 by as much as 3:1 in some key applications [8]. Improved coding efficiency comes with the added cost of complexity in the encoding and decoding processes. H.264 utilizes some methods like multiplier-free integer transforms and simple integer based shift and add arithmetic that are ASIC friendly and reduce implementation complexity. Applications like digital video set-top box, handheld devices and HD-DVD players require portable, low power ASIC video decoders. ASIC implementation of H.264 decoder is a challenge because of complex algorithms and various modes. Functional verification of the ASIC H.264

decoder is more complicated and generating test patterns to verify all modes and corner cases is difficult. This chapter gives an overview of H.264, its ASIC implementation and explores verification challenges.

2.1 Overview of H.264

H.264 standard has been well segregated into profiles and levels to meet the requirements of various video related applications. The encoding process can be represented by Figure 2-1 [4].

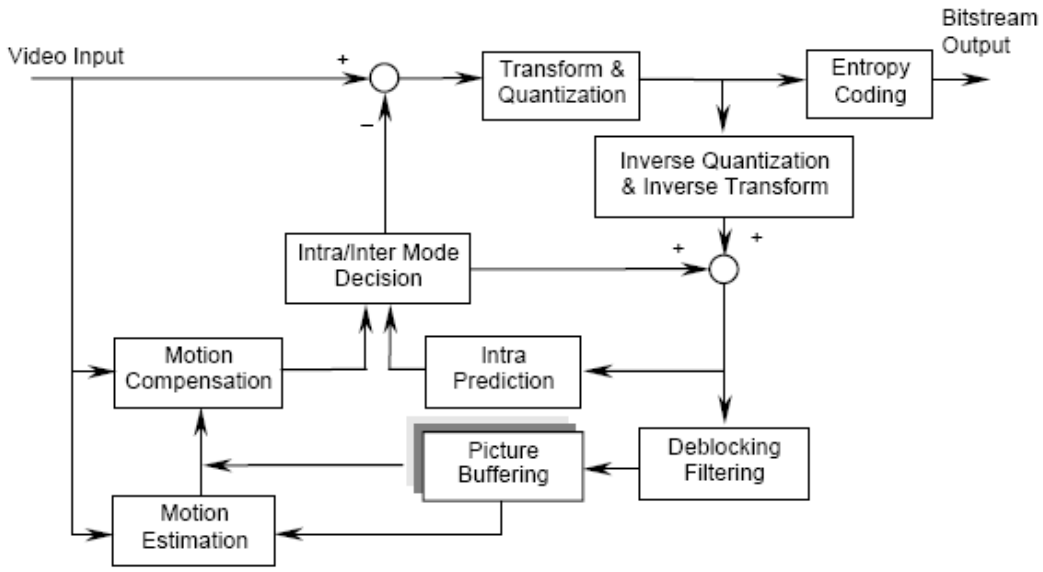

Figure 2- 1: H.264 encoder block diagram [4]

H.264 uses integer DCT, quantization and intra prediction techniques to reduce spatial correlation. Integer DCT is employed to reduce implementation complexity and avoid encoder decoder mismatch. Temporal redundancy is reduced by motion estimation based inter prediction techniques. Compression efficiency and visual quality are enhanced by using various block sizes and up to quarter pixel accuracy for motion

estimation. Statistical redundancy is minimized using variable length coding (VLC). Advanced coding techniques such as exponential Golomb encoding, Context-based Adaptive VLC (CAVLC) and Context-based Adaptive Binary Arithmetic Coding (CABAC) are employed to increase compression efficiency. Coding of video is done one frame/picture at a time and each picture is divided into one or more slices. A slice is the smallest completely decodable element in an H.264 video stream. Each slice is a sequence of macroblocks with each macroblock consisting of 16X16 pixels. Encoded slice data and other required information for decoding a slice are packed into a Network Abstraction Layer Unit (NALU). A sequence of these NALU makes a H.264 video stream. Overview of the standard and algorithms are presented in the following sections.

2.1.1 Profiles and levels

H.264 defines seven profiles that address different video related applications from low bandwidth networks to digital cinema. All profiles in H.264 have some common coding parts and some specific coding parts as shown in Figure 2-2 [4].

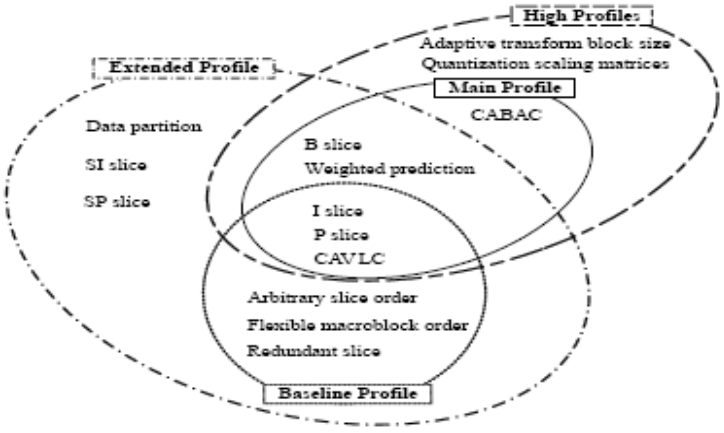

Figure 2- 2: Specific coding parts of the profiles in H.264 [4]

Some common features to all profiles are:

1. Intra-coded slices (I slice): These slices are coded using prediction only from decoded samples within the same slice.
2. Predictive-coded slices (P slice): These slices are coded using inter-prediction from previously decoded reference pictures. Sample values of each block are predicted using one motion vector and reference index.
3. 4X4 modified integer DCT
4. CAVLC for entropy encoding
5. Exponential Golomb encoding for headers and associated slice data

2.1.1.1 Baseline Profile

Baseline profile mainly addresses the real time video applications such as video conferencing, video phone and incorporates tools that help in error resilience. Simpler algorithms are used in prediction to keep the decoding process less complex and less memory intensive. Some salient features are:

1. I and P slice only.
2. Only CAVLC based entropy encoding.
3. Flexible Macroblock Ordering (FMO), macroblocks need not be presented to the decoder in raster scan order.
4. Arbitrary Slice Ordering (ASO), slices need not be presented to the decoder in raster scan order.

5. Redundant Slice, coded slice data of the previously coded slice at same or different coding rate.

First two make the decoding process simple and the later three help to achieve error resilience.

2.1.1.2 Main Profile

Features in main profile are designed to best suit the digital storage media, television broadcasting and set-top box applications. It uses more sophisticated prediction and entropy coding techniques and eliminates error resilience tools. Some important features are:

1. I, P and B slices are supported. B slices (Bi-directionally predictive-coded slices) are slices coded using inter-prediction from previously decoded reference pictures using at most 2 motion vectors and reference indices.
2. Both CAVLC and CABAC based entropy coding is supported.
3. Weighted prediction, a technique of scaling the samples of motion-compensated prediction data by a weighting factor can be used to give better prediction for fading scenes.

2.1.1.3 Extended profile

Extended profile caters to the needs of multimedia services over internet like streaming video and video on demand. This profile uses the error resilience tools from baseline and prediction tools from main profile and adds more features like data

partitioning and switching slices that are best suited for streaming applications. Some significant features are:

1. I, P and B slices are supported.
2. Weighted prediction is supported
3. All error resilience tools, FMO, ASO and redundant slices are supported
4. Switching I and P slices (SI, SP Slices) are introduced for efficient switching between video streams.

2.1.1.4 High profiles

H.264 also defines four high profiles [8] which are more sophisticated and cater applications involving high quality and high resolution video such as studio editing, content distribution, post processing and digital cinema. All high profiles have some common and some specific coding tools and features. Common features of high profile include:

1. All main profile features such as I, P and B slices, weighted prediction and CABAC.
2. Adaptive transform block size. Both 4X4 and 8X8 block sizes are supported
3. Perceptual quantization matrices to improve subjective quality

Additionally, High 10 profile supports pixel bit-depths up to 10 bits. High 4:2:2 profile supports 4:2:2 chroma sampling and up to 10 bits per pixel. Advanced 4:4:4 profile supports 4:4:4 chroma sampling and up to 12 bits per pixel.

2.1.1.5 Levels

For all profiles, level specifies the limitation on frame size (number of macroblocks per frame), frame rate, memory allocations, bit rates, motion vector search range and minimum compression ratio. Table 2-1 [2] details the level specifications

Table 2-1: Level specifications in H.264 [2]

Level number	Max macroblock processing rate MaxMBPS (MB/s)	Max frame size MaxFS (MBs)	Max decoded picture buffer size MaxDPB (1024 bytes for 4:2:0)	Max video bit rate MaxBR (1000 bits/s, 1200 bits/s, cpbBrVclFactor bits/s, or cpbBrNalFactor bits/s)	Max CPB size MaxCPB (1000 bits, 1200 bits, cpbBrVclFactor bits, or cpbBrNalFactor bits)	Vertical MV component range MaxVmvR (luma frame samples)	Min compression ratio MinCR	Max number of motion vectors per two consecutive MBs MaxMvsPer2Mb
1	1 485	99	148.5	64	175	[-64,+63.75]	2	-
1b	1 485	99	148.5	128	350	[-64,+63.75]	2	-
1.1	3 000	396	337.5	192	500	[-128,+127.75]	2	-
1.2	6 000	396	891.0	384	1 000	[-128,+127.75]	2	-
1.3	11 880	396	891.0	768	2 000	[-128,+127.75]	2	-
2	11 880	396	891.0	2 000	2 000	[-128,+127.75]	2	-
2.1	19 800	792	1 782.0	4 000	4 000	[-256,+255.75]	2	-
2.2	20 250	1 620	3 037.5	4 000	4 000	[-256,+255.75]	2	-
3	40 500	1 620	3 037.5	10 000	10 000	[-256,+255.75]	2	32
3.1	108 000	3 600	6 750.0	14 000	14 000	[-512,+511.75]	4	16
3.2	216 000	5 120	7 680.0	20 000	20 000	[-512,+511.75]	4	16
4	245 760	8 192	12 288.0	20 000	25 000	[-512,+511.75]	4	16
4.1	245 760	8 192	12 288.0	50 000	62 500	[-512,+511.75]	2	16
4.2	522 240	8 704	13 056.0	50 000	62 500	[-512,+511.75]	2	16
5	589 824	22 080	41 400.0	135 000	135 000	[-512,+511.75]	2	16
5.1	983 040	36 864	69 120.0	240 000	240 000	[-512,+511.75]	2	16

2.1.2 Video decoding process

H.264 standard gives the bit stream syntax and details the decoding process and algorithm, to decode a coded video stream. For the encoder, it specifies bit stream syntax, normative and informative guidelines to generate a compliant encoded

sequence. The decoding process can be understood with the help of a block diagram shown in Figure 2-3. Important algorithms are briefly explained in the following sections with emphasis on baseline profile decoding.

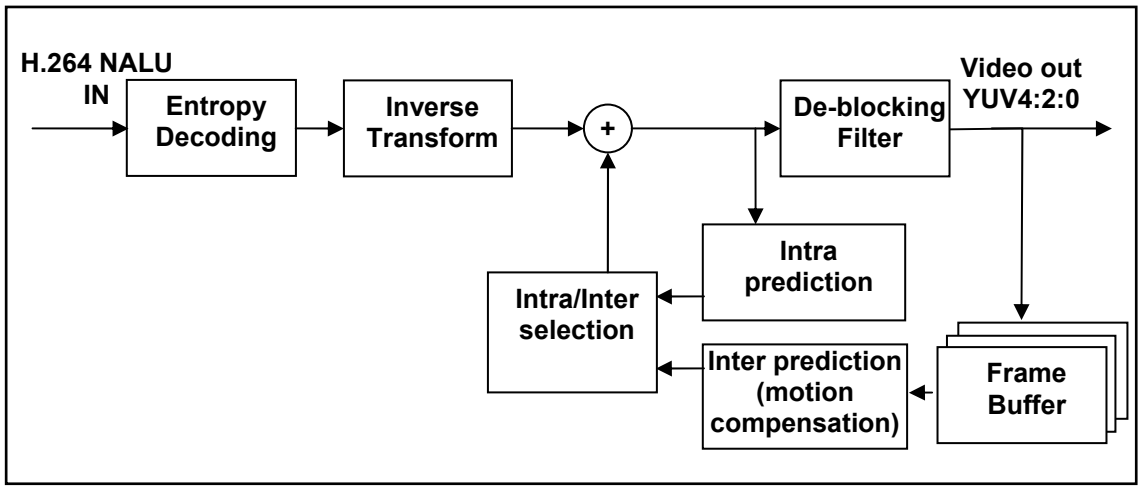

Figure 2- 3: Functional Block diagram of an H.264 decoder

2.1.2.1 Entropy decoding

In Baseline profile H.264 uses CAVLC based entropy coding for slice data encoding and exponential Golomb coding for syntax elements and header information [9]. Role of the bit stream parser or the entropy decoding block is

1. To identify NALU start headers.
2. Decode parameter sets and update relevant decoder parameters.
3. Decode slice header information and update syntax element parameters.
4. Perform CAVLC based variable length decoding. Inputs to this process are bits from slice data and maximum number of non-zero transform coefficient levels in a 4X4 block. Output of this process is 4X4 block of transform coefficient levels.

5. Perform inverse zigzag scan ordering of data to recover transformed residual block.

2.1.2.2 Inverse transform

H.264 uses modified 4X4 integer DCT to avoid mismatch in encoder and decoder [10]. Scaling multiplication of transformation is integrated into the quantization process [10]. H.264 further exploits correlation in 16 DC values of the transformed macroblock by applying 4X4 Hadamard transform on Luma DC coefficients and 2X2 Hadamard transform on Chroma DC components [11]. Input to this block is entropy decoded and inverse zigzag scanned macroblocks and output of this block is inverse quantized and inverse transformed macroblocks. Inverse transform block has to perform

1. Inverse quantization
2. Inverse Hadamard transform for luma and chroma DC coefficients.
3. Inverse DCT of 4X4 blocks.

2.1.2.3 Intra prediction

H.264 exploits spatial correlation in intra coded macroblocks with intra-prediction techniques. Inputs to this process are reconstructed macroblocks, intra prediction modes of current macroblock and previously decoded neighboring macroblocks. Output of this block is constructed samples prior to de-blocking filter. H.264 uses 9 modes of intra-prediction on 4X4 blocks and 8X8 blocks and 4 modes on

16X16 blocks. The modes and their prediction type are pictorially represented in Figure 2-4 [12].

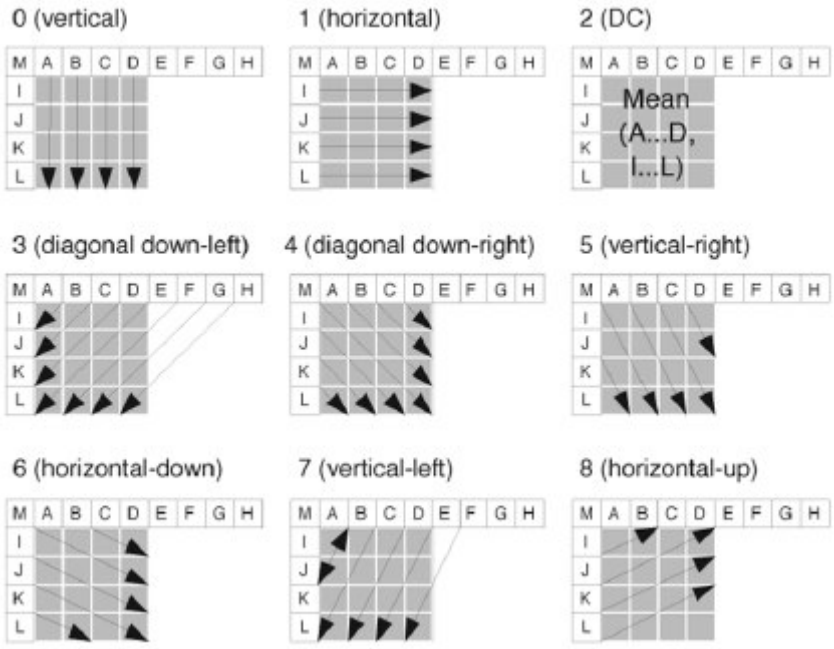

Figure 2- 4: Pictorial representation of intra prediction modes in H.264 [12]

2.1.2.4 Inter prediction

Inter prediction is used to take advantage of temporal redundancy in video data. In general, a large partition size is appropriate for homogeneous areas of the frame and small partition size is beneficial for areas with more details [4]. H.264 gives provision to use block sizes ranging from 16X16 up to 4X4. Sub-pixel motion compensation, with added complexity, outperforms integer-pixel motion compensation in terms of compression efficiency [13]. H.264 supports up to quarter pixel accuracy of motion compensation. Half-pixels and quarter-pixels are derived from full-pixels using 6-tap

FIR and bi-linear filters. Inputs to this block are motion vectors and reconstructed residual data. Output of this block is a predicted macroblock.

2.1.2.5 De-blocking filter

To reduce the blocking artifacts introduced by block-based transform, inter and intra predictions and quantization, H.264 employs an adaptive and optional in loop de-blocking filter. De-blocking filter has significant impact on visual quality improvement [14]. Filtering is applied adaptively along the 4X4 block edges. Input to this block is a completely reconstructed macroblock, boundary strength and quantization parameters. Outputs of this block are the final reconstructed macroblocks.

2.2 ASIC implementation

With the advent of digital video and audio and supporting technologies like flat panel displays and compact speakers, multimedia based consumer products have become very popular. Digital multimedia is now part of all personal entertainment systems, car entertainment systems, TV broadcasting, video conferencing, video on demand and mobile TV to name a few. Key feature of digital video is its compression technology like H.264, because of which storage and transfer are compact and cost effective. All the end-user products inevitably require a cost effective decompression solution that is compact in size and consumes less power. Low power, low cost digital signal processors do not have enough processing power to decode complex algorithms used in H.264 [12]. These requirements and volume of the consumer electronics market

encourage ASIC implementation of H.264 decoder. Following sections give an overview of the ASIC implementation of H.264 decoder.

2.2.1 Hardware Description Language

VHDL and Verilog are the two commonly used Hardware Description Languages (HDL) in the VLSI industry. In this thesis, VHDL is used for implementing the ASIC H.264 decoder core. VHDL stands for VHSIC Hardware Description Language. VHSIC is itself an abbreviation for Very High Speed Integrated Circuits, an initiative funded by the United States Department of Defense in the 1980s that led to the creation of VHDL [15]. VHDL was the original and first hardware description language to be standardized by the IEEE, through the IEEE 1076 [16] standard in 1988. An additional standard, the IEEE 1164 [17], was later added to introduce a multi-valued logic system in 1993. The IEEE 1076 was further revised in 2000 and the current revision was updated in 2002 [18]. A hardware design can be implemented using VHDL in a descriptive structure and the code can be used for hardware simulation and synthesis. Benefits of using VHDL are:

1. VHDL is a standard
2. It is a technology or vendor independent language
3. It is easily portable and reusable
4. Modular level design and system integration are easy
5. It is supported by both FPGA vendors and ASIC foundries for fabrication

2.2.2 ASIC design process

Goal of the ASIC H.264 decoder design is to achieve real-time decoding (up to 30 frames per second) of H.264 compressed high definition (HD) video of resolution up to 1280X720 (720p) with an operating frequency less than 100 MHz. Guidelines for the design process are:

1. Compliance to the standard
2. Complete hardware implementation
3. Target or platform independent implementation
4. Common code base for ASIC and FPGA prototyping

H.264 standard was thoroughly studied to understand the requirements and specifications of the standard. The decoder design is split into modules and each module represents a block in Figure 2-5. Modules are instantiated as components in the decoder

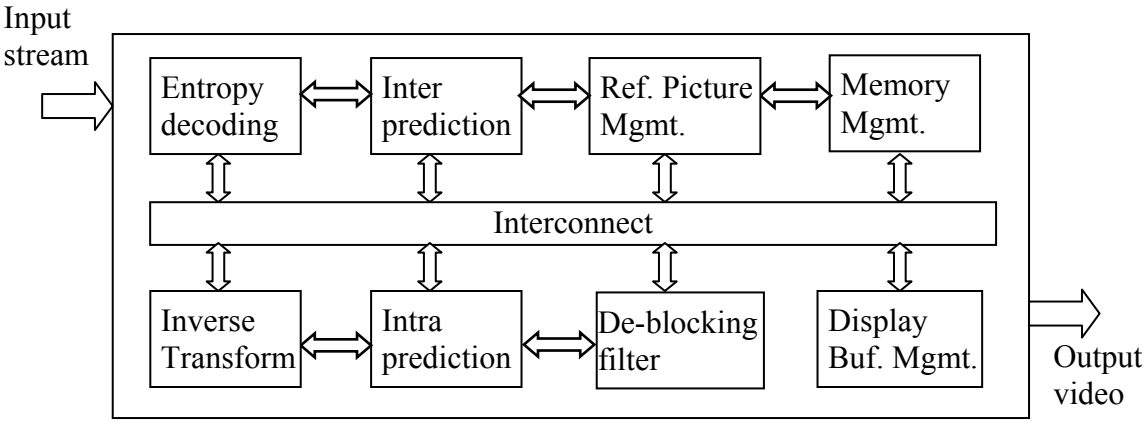

Figure 2- 5: ASIC modules of H.264 decoder core

and required interconnects are made between the modules to pass data and control signals. Decoder design employs a pipelined architecture to makes use of concurrency in operation of an ASIC. Each module processes one macroblock at a time independent

of the other modules. Functional description of most of the modules is described in section 2.1.2. Each module gets the data and other parameters for processing from input memory buffers and stores its processed data and parameters in output memory buffers. Memory allocations and read and write requests of each module are handled by memory management module. It performs proper allocations and arbitration of requests and provides single memory interface. Reference picture management module works closely with inter-prediction to provide the required reference picture and to store decoded frames that can be used as reference pictures in future. Display buffer management module outputs the decoded frames in display order.

Design process involves understanding the specifications of H.264 baseline profile decoder, partitioning the design into modules, coding of individual modules in VHDL and testing for proper functionality. Individual modules were simulated with test vectors passed from the test-benches and verified with expected values. Functionally verified modules were integrated and interconnected to form the decoder. H.264 compressed bit stream was passed as test vector to simulate and functionally verify the decoder.

2.3 Summary

In this chapter, overview of H.264 video coding standard was presented and compression algorithms employed were briefly explained in sub-sections of 2.1. Sections of 2.2 elaborated on need for ASIC H.264 video decoder and also discussed the design process employed. Functional verification methodologies and FPGA prototyping based functional verification are discussed in the following chapter.

CHAPTER 3

FUNCTIONAL VERIFICATION OF ASIC DESIGN

With increases in complexity and gate count of an ASIC design, functional verification has become one of the greatest concerns of design engineers. Verification has also become a serious bottleneck in the VLSI design process. In [19], Dhodhi et al. estimate that functional verification takes 50-60 % of time and efforts of design teams. VLSI design methodologies that take into account verification issues in the early phase of design and utilize state-of-the-art techniques have become a necessity. With technology changing more often than ever, time to market has become crucial for all design houses and manufacturing companies. Sections in this chapter present functional verification techniques currently in practice in the industry and discuss their advantages and limitations. In this chapter, FPGA prototyping based functional verification is introduced as a fast and efficient alternative for functional verification.

3.1 Functional verification

Engineering design process is not complete without the verification of the design. Verification is a process of checking whether the design meets the specifications for which it was designed. Verification of an ASIC can be broadly classified into timing verification and functional verification. In timing verification, design is tested against the timing requirements of the system such as frequency of operation, meeting set-up

and hold times etc. Functional verification is a process in which, logical or functional correctness of a digital circuit is verified [20]. In functional verification response or output of the digital circuit is matched against its expected output for a particular set of stimuli or inputs. ASIC design process can be represented with the flow diagram shown in Figure 3-1.

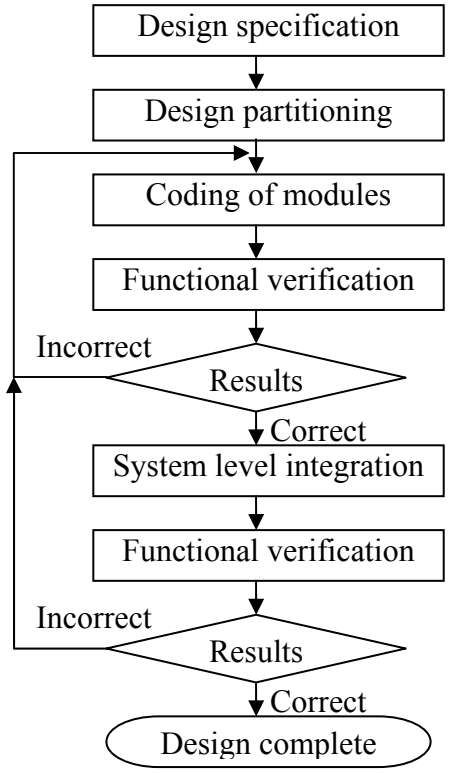

Figure 3- 1: ASIC design process

Coding and functional verification is an iterative process where verification consumes most of the time as the design has to be verified thoroughly even for minor modifications or corrections made in the code. Efforts involved and the time consumed

at system level verification are the highest and are of major concern [20]. Two most important components of functional verification are speed of functional execution and design controllability and observability. There are various tools and techniques that are currently in practice in the industry that are categorized under Electronic Design Automation (EDA) [21].

3.2 Verification by software simulation

In software simulation based verification, the HDL code of the digital logic is simulated by the simulation software. Logic simulation is the primary tool used for verifying the logical correctness of a hardware design. In many cases, logic simulation is the first activity performed in the process of taking a hardware design from concept to realization. Test-bench can be written around the design under test (DUT) and inputs can be passed to the DUT through the test-bench. VHDL provides synthesizable and non-synthesizable constructs. Test-bench can use constructs like file I/O and pre-defined patterns to generate inputs to the DUT. The outputs are automatically verified and reported using report or assert statements. Outputs can also be written to a file with file I/O for manual or automated verification. Simulation software compiles the DUT and uses the test vectors provided by test-bench to simulate the design. Simulation software also keeps track of all the signals and ports in the design and their transition at the change of the clock or inputs and writes it to a waveform database. Waveform viewer software is used to analyze the performance of the digital design and understand the cause or source of the error if any. Passing the simulation test can reasonably assure

the logical correctness of the design, for the cases that have been tested in the simulation.

3.2.1 Advantages and limitations of logic simulation

Logic simulation comes with following advantages

1. Simulation is completely generic and any hardware design can be simulated.
2. Setup is simple, quick and easy
3. Highest level of controllability and observability
4. Designer gets complete feedback of the verification process
5. Direct interaction with the design with minimum abstraction, there is no layer of translation to obscure the behavior of the design
6. Changes in every signal or port can be observed for every change in the input during and after the verification process through waveforms.
7. No additional hardaware cost or porting efforts.

The ease of software simulation based verification comes with an overhead of simulation time which increases with the complextiy of the design. As discussed in [3, 22, 23, 24], time consumed in simulating a digital design is a major drawback of simulation based verification. The time required to verify the design is proportional to the maturity of the design. Early in the design process at module level verification, incorrect functionalities are usually found quickly and easily through simulation. As the design matures, it takes longer to find the errors. Simulation time largely varies

depending on the software used, computer configuration and coding style. In [22], Tschäche and Sieh as an example, estimate the simulation execution time per gate as 2.9 ms. These estimations are for verification of a simple 2 input NAND gate. Further in [25], Baker and Mahmood present a generalized mathematical model for time of unit delay simulation shown in (3-1) which is derived from Bailey's mathematical model [26].

$$\tau_{s,u} = \sum_{i=0}^{n-1} E * \left(depth(\beta_i) + 1\right) * e_i \qquad (3\text{-}1)$$

where
$\tau_{s,u}$ - Time of unit delay simulation
E - Evaluation time per level
β_i - Simulation dependency graph for an input event
e_i - Input event
n - Number of input events

For simple designs all possible input cases can be exhaustively passed through the design and the output can be verified. As the complexity of the ASIC design increases depth of simulation dependency graph increases and also the number of input events increases drastically. With the increase in design size, generating and testing all possible test vectors becomes difficult and time consuming. For larger designs not only the exhaustiveness of input test vectors but also the order in which they are passed decides the behavior of the logic thus making the list of exhaustive test vectors infinitely long. Especially, when the designs include signal processing algorithms like video processing with various conditional modes and functionality the order of test vectors passed is crucial. For instance a third frame predicted using the second frame

cannot be decoded until the first two frames are completely decoded. Also testing the individual modules exhaustively is not possible because of the dependency involved. In H.264, decoding of current macroblock requires decoded data from previous macroblocks and previous modules in fixed order. For instance, to perform inter-prediction on a particular macroblock, motion vectors of neighboring macroblocks and data from previous frame are required. Writing test-bench codes to generate such input patterns is very time consuming and sometimes the code volume of the test-bench is more than the design code itself [20].

Apart from being time consuming, simulation based verification also has other drawbacks. Simulation can take an inordinately large amount of computing resources, as it typically uses a single processor to reproduce the behavior of many (perhaps millions of) parallel hardware processes. Also memory requirements of simulation tool increase with the complexity of the design and become prominent when the design and test-bench have bigger memory instantiations. For instance, simulation tool reports insufficient memory for decoding a high definition sequence with a resolution of 1280X720. Some experiments were performed to measure the performance of simulation tool. NC-VHDL simulation software [27] from Cadence Design Systems was used on a Pentium 4 system running at 3.2 GHz with 1GB of DDR2 primary memory. Table 3-1 lists the gate count of some modules in H.264 decoder and the average number of clocks required to complete the processing of one macroblock. Last column lists the average time taken by the simulator to simulate the design. The inverse transform module running at 50 MHz in real-time would require 65 μs to process a

block, but a simulator takes 1.14 seconds. Despite simulator being more than 17,000 times slower than real-time, 1 second of simulation time is acceptable for the ease, controllability and observability the simulation tool provides for the designer. These features are more important during the initial design process.

Table 3- 1: Gate count and simulation time of H.264 modules

SL. No.	H.264 decoder Module	Gate count (K)	Avg. no. of clocks per macro-block	Avg. Simulation time (s)
1	Inverse transform	95.57	3280	1.14
2	Intra prediction	99.43	3268	1.3
3	De-blocking	87.24	3093	1.1
4	Inter prediction	170.31	3025	1.4

Table 3- 2: Simulation time of an ASIC H.264 decoder and memory requirements

SL. No.	Frame resolution of the sequences decoded	Avg. no. of clocks per frame	Avg. simulation time	Avg. Memory utilization (MB)
1	QCIF (176X144)	279848	1 min,10 sec	946
2	CIF (352X288)	978958	5 min, 42 sec	938
3	4CIF (704X576)	4054708	27 min,50 sec	1089

Table 3-2 lists the average number of clocks required for the H.264 decoder to decode compressed streams of different frame resolutions. Columns 4 and 5 list the average simulation time and the system memory used by the simulator to simulate the H.264 decoder with approximatley 500 K gate count. Simulator on an average takes 342 seconds to decode one frame of a CIF resolution sequence. At 30 frames per second the simulator would take approximately 11 and ½ hour to simulate decoding of 4 seconds

of a CIF sequence and more than two days to simulate decoding of a 4CIF sequence. Such long delays make simulation based verification inappropriate for thorough functional verification of ASIC H.264 decoder.

Considering both the advantages and disadvantages of logic simulation, it is a good tool for verifying the correctness of a hardware design and is a tool of choice for module level verification and verification of small designs. But it has major drawbacks for verification of large complex design like video codec.

3.3 Emulation based verification

Emulation based verification is a faster verification tool compared to software simulation based verfication. In [28], Nguyen and Thill discuss the advantages of using emulation based verification over software simulation based functional verification. In [29] Walters highlights the impractical limitations of simulation based verification and proposes emulation based verification as an alternative for thorough verification of ASIC designs. Typically, emulation based verfication tools come with a hardware accelerator card that helps to speed-up simulation and a software program that interfaces the card and the software simulation tools. The hardware accelerator card has one or more programmable devices and a set of fixed interfaces. The software takes the HDL code of DUT and partitions it to fit into the programmable devices on the card. Also signals and/or ports that need to be monitored are taken as input from the user and extra logic to monitor them is programmed on to the accelerator card. Once the card is ready, inputs are taken from the test-bench through the simulation software and passed

on to the card. The ouputs produced on the ports being monitored are sent back from the accelerator card to the simulation software. These signals or ports can then be analyzed automatically or manually. The emulation tools make use of the program language interface (PLI) constructs of verilog or foreign language interface (FLI) constructs of VHDL to interact with the test-bench which are in verilog or VHDL. Some emulation tools use custom languages to generate test-benches. The emulation based verfication process is represented in Figure 3-2.

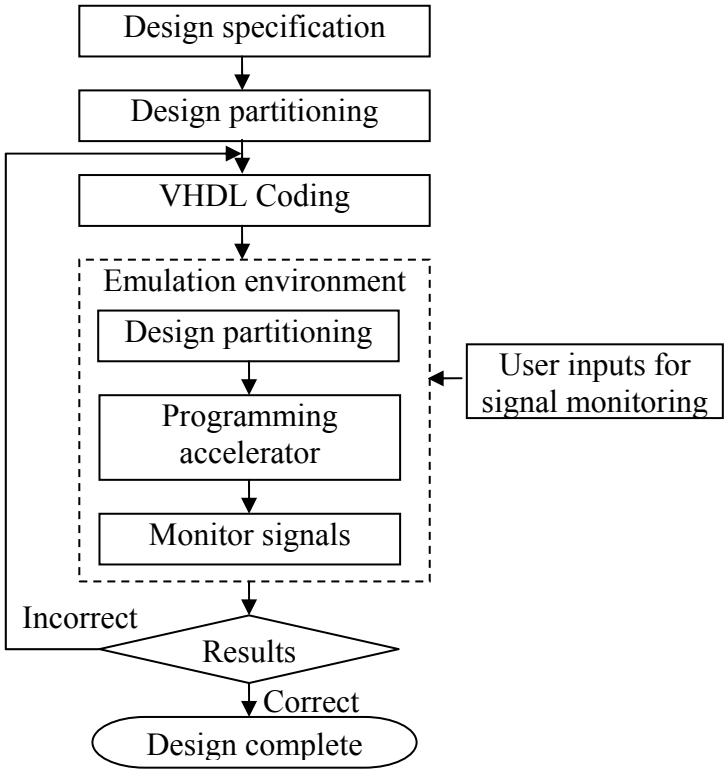

Figure 3- 2: Emulation based verification process

Many EDA tool vendors provide emulation based verification tools as a faster alternative to simulation based verification. An emulation tool like VStationPRO from Mentor Graphics coporation [30] or Palladium from Cadence Design Systems Inc [31]

provides 100-10,000X faster performance compared to software simulation. These emulation platforms are scalable and can verify more than 100 million gates at speeds 500 – 2000 kHz. Today, automated and advanced emulation tools are available to the designers. The emulation environment is kept at a level of abstraction to the user and the software gives an easy interface for verifying the DUT. The emulation software takes the RTL and test-bench of the DUT, automatically partitions the DUT and programs the accelrator to provide all the data required by the test bench for verification. It uses PLI and FLI constructs to interact and exchange data with the accelerator and the simulaiton software. The designer can access and monitor the requested signals and the test-bench can automatically verify the results against the expected values.

3.3.1 Advantages and limitations of emulation

Emulation based verification has following advantages

1. Verification is 100 to 10000 times faster than simulation.
2. Emulation gives controllability and observability with certain level of abstraction
3. Partial to complete feedback from the verification process
4. Changes in desired signals or ports can be observed for every change in the input during and after the verification process through the waveforms.
5. Can simulate multi-million gate ASIC design.
6. Emulators are scalable and have various standard interfaces and built-in bus functional models (BFM)

Speed improvement in verification through emulation comes with an overhead of extra cost for the emulator hardware and design time. Hardware emulation platforms can cost up to a million dollars [32] and can run only at a speed of 1 or 2 MHz. This speed is 100 to 1000 times faster than simulation but still too slow for some applications like video processing. The speed of operation and the number of ports or signals that can be observed reduces with the increase in the design complexity. Increase in design complexity also increases the time required by compiler tools to partition the design and generate programmable code for the devices on the accelerator card. Learning curve in understanding the accelerator platform and its capabalities is usually long. If the emulation platform uses its own language and syntax structure, modification of the design and test-bench becomes an overhead. Accelerator platforms are more generic in nature having wide varitey of interfaces and IP cores and not customized for specific designs making them expensive. Also the emulator platforms do not run real time and cannot generate proper signals to time sensitive devices like memory contollers or video screens. So the interfaces cannot be verified in real-time.

Considering the advantages and disadvantages, emulation based verification is a faster alternative compared to software simulation but it is very expensive. Also the speeds achieved do not give real-time performance and are much slower than expected for many applications. ASIC designs like video codec require faster and cost-effective verification tecniques to reduce time to market and design cost.

3.4 FPGA prototyping based verification

Field programmable gate array (FPGA) is a semiconductor device containing programmable logic blocks and programmable switches that interconnect the logic blocks. Also, FPGA are reconfigurable [33]. These features of FPGA allow them to be used for any application and quick prototyping. ASIC designs are generally time consuming and are not cost effective for small designs and low volume production. FPGAs were introduced as an alternative to ASIC to shorten the time to market and overcome huge production cost of ASIC for small designs. As the gate density on the FPGA increased, they were quickly adopted into emulation based verification tools to significantly enhance the speed of simulation based verification techniques. FPGAs are also used to prototype a fully verified ASIC design for system level co-verification on custom designed boards before going for actual fabrication [3]. Many approaches and techniques for system level hardware software co-verification using FPGA have been suggested in [34,35,36].

With the increased use of FPGA for prototyping of an ASIC, many vendors now provide custom off the shelf boards with application specific interfaces for rapid prototyping. The cost of these boards mainly depends on the density of the FPGA and the interfaces present and vary from few hundred dollars to several thousand dollars. Current FPGA technology can accommodate very large complex designs and can run up to speeds of 500 MHz [37]. These rapid prototyping boards with appropriate FPGA and interfaces can run ASIC designs in real-time. Speed of operation and low cost make them a better alternative for verification as compared to simulation or emulation based

verification. In this thesis an approach of incorporating FPGA prototyping into the ASIC design process for faster design verification of ASIC H.264 decoder is adopted and developed. Chapter 4 elaborates the FPGA prototyping based verification process and its advantages and limitations over earlier techniques.

3.5 Summary

In this chapter, functional verification of ASIC design is discussed and the techniques currently in practice in the industry are presented. The limitations of simulation based verification and emulation based verification for verifying ASIC H.264 decoders are discussed. To overcome time consumed by simulation software and expenses of emulation techniques, FPGA prototyping based verification is suggested as a faster and low cost alternative. The following chapter will elaborate on how FPGA prototyping is used for verification of ASIC H.264 video codec and the gains of this technique.

CHAPTER 4

FPGA PROTOTYPING AND FUNCTIONAL VERIFICATION

Since its introduction in mid 1980s [38], FPGA technology has advanced many folds and today it makes use of the best VLSI technology available to provide high capacity high performance programmable devices. State of the art FPGAs are the largest integrated circuits manufactured [39]. The reconfigurability of FPGA has made them a faster and economical choice for prototyping and system level testing and verification. Also, custom off the shelf prototyping boards with desired interfaces are easily available and are cost effective. Design verification time can be significantly reduced if we make use of FPGA prototyping boards earlier in the ASIC design process for functional verification. This technique gives substantial increase in speed over simulation and emulation based verifications and is a low cost alternative. In this chapter, FPGA, prototyping boards and their selection criteria are discussed. Verification strategy employed and the improvements achieved in terms of performance are presented. Also, merits and demerits of FPGA prototyping based verification are highlighted and new approaches are suggested.

4.1 FPGA

FPGA [33] contain programmable logic cells and programmable switches that interconnect the logic cells. FPGA are reconfigurable and can be used to prototype any ASIC design. FPGA come with different capacities and programmable functional blocks. Criteria for selecting a right FPGA that best suits for the design to be prototyped include

1. Capacity of the FPGA. Each configurable logic cell in the FPGA can be programmed to replicate the actual gates in the ASIC design. Depending on the design logic, a logic cell can be completely utilized, partially utilized or just used for routing purposes. Considering all this, the FPGA with enough logic cells to accommodate the design should be selected.
2. Speed of operation. Speed of operation is critical to get the best or expected real time performance from the prototype. The operating frequency of individual programming blocks and the interconnect delay play an important role to decide the frequency of operation of the prototype.
3. Logic resources. Depending on the complexity of the design, the FPGA should have enough functional logic blocks such as memory, clock dividers, DSP blocks, multipliers, comparators, arithmetic blocks and input/output ports.
4. Synthesis and implementation tools. Synthesis tools compile and translate the design written in VHDL code into FPGA specific code. Implementation tools perform the place and route operation to fit the logic into the FPGA

and choose appropriate logic cells to replicate the functionality. Efficient synthesis and implementation tools make best use of the logic resources on the FPGA and help to provide best speed performance.

5. IP cores and interface logic. The performance of the prototype can be improved if the FPGA vendors provide optimized implementation of commonly used cores like memory modules, memory controllers, clock dividers, and standard bus interfaces specific to the FPGA. It will also reduce design and verification time of these modules.

All these factors are considered and Virtex-4LX100 [40] FPGA from Xilinx Inc. [41] is used to prototype the ASIC H.264 decoder. The logic blocks in a Virtex-4 FPGA can operate at frequencies up to 500 MHz [42]. Virtex-4LX100 has 100,000 logic cells and assuming typical gate count as mentioned in [43] to be 12 per logic cell, it can hold up to a million ASIC gates. Virtex-4 FPGA offer DSP specific blocks that are high performance versatile arithmetic units. These can also be configured as counters, shift registers and accumulators [42]. Virtex-4LX100 provides 4,320 Kbits of block RAM and high performance external memory interface [40]. Xilinx also provides Integrated Synthesis Environment (ISE) tool [44], which performs synthesis and implementation. ISE also comes with core generator tool that can be used to generate standard IP cores like memory modules, interfaces and DSP blocks optimized for the specific FPGA. Considering all the advantages Virtex-4LX100 is the best choice to prototype H.264 decoder design.

4.2 Prototyping board

The important criteria for choosing an appropriate board for prototyping an ASIC design are

1. Interfaces present on the board. Whether the board has all the required interfaces and expandability to add more functionality is an important requirement for prototyping. For video decoder designs, the board should have interface to support streaming of the compressed video sequence, fast and sufficient memory to store decoded frame data and other intermediate data. Other important interface required is the display interface to display decoded frames on to the monitor.
2. Reusability for other projects. The board should not be an obsolete design and should have advanced peripheral devices and interfaces that can be used for other related projects. This will divide the burden of cost of the board among other projects.
3. Cost of the board. Cost of the board is an overhead to the project design cost. The design of the board should have well balance of general features for reusability and project specific features for cost effectiveness and optimized performance.
4. Learning time. Time required for understanding and learning the peripheral interfaces and associated software of the board is an overhead on project design time. Boards with simple and well defined standard interfaces require

less time to learn. Also if proper reference designs and software with appropriate drivers are supplied by board vendors, it reduces learning time.

5. Software and hardware requirements. Any additional software or hardware design other than those provided by board vendors consumes lots of time and resource. Especially designing customized hardware consumes lots of time and such design should be avoided.
6. Data transfer rate. This requirement is more specific to verification than prototyping. To monitor any signals or values at any modules, data needs to be transferred to the computer for automated or manual verification. Data transfer rate decides the performance of the prototype and the number of signals or ports that can be monitored.

The prototyping board chosen considering all the above factors is the DN8000K10PCI [45] board from the Dini Group [46]. The DN8000K10PCI board can host up to 2 Virtex-4LX100 or Virtex-4LX200 and a Virtex-4FX60 or Virtex-4FX100 Xilinx FPGAs. The board provides two slots of DDR2 memory with addressing capacity of up to 4GB in each slot for large data storage and fast access. The board has serial port, PCI port [47] and USB port [48] for data exchange and monitoring from a computer. The board also provides programmable clocks to generate clock signals for desired frequency of operation. Board also has two 200-pin expansion connectors to connect daughter cards or other peripheral hardware. The board is supplied with PCI core implementation, USB drivers, relevant software, connecting cables and reference designs. Dini group also offer Digital Video Interface (DVI) daughter-card [49] that

connects to the expansion connector of DN8000K10PCI and provides display interfaces. The DN800K10PCI and DVI card pair meets all the requirements for verifying and prototyping the ASIC H.264 decoder and the solution is cost effective. Figure 4-1 shows the DN8000K10PCI card used for prototyping ASIC H.264 decoder and Figure 4-2 shows the DVI daughter-card used for displaying the decoded video.

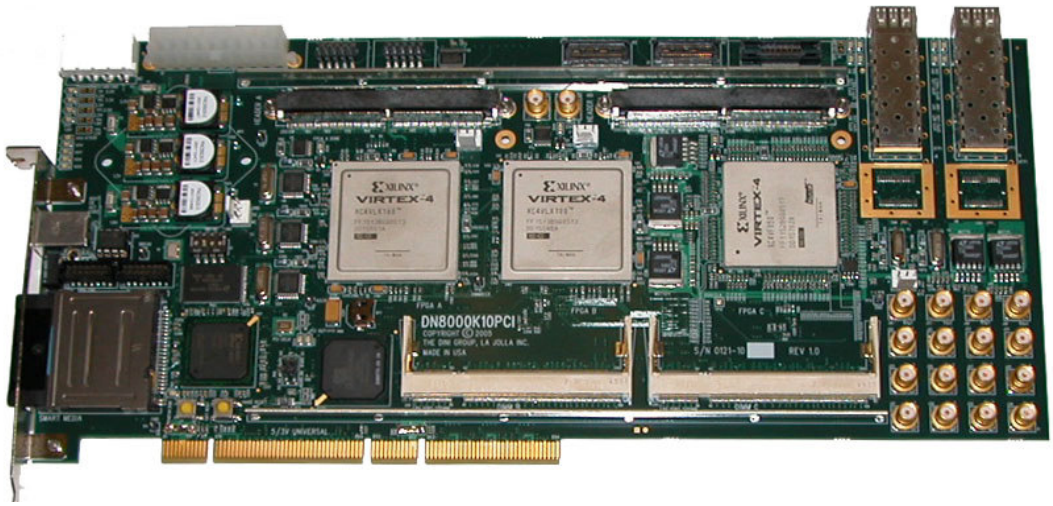

Figure 4- 1: Prototyping board used for verifying ASIC H.264 decoder [45]

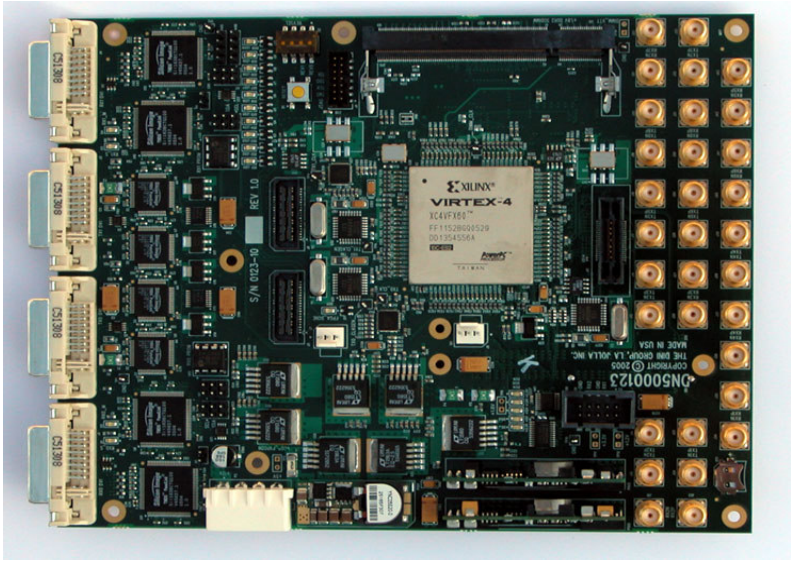

Figure 4- 2: Display card used with prototyping board for displaying decoded video [49]

4.3 Prototyping based verification strategy

To overcome the huge delay in simulation based verification, FPGA prototyping based verification technique is used. After completion of module level coding and verification for handshake and data arrival at module level using simulation tool, the modules were integrated to form the complete H.264 decoder core. While initial simulations were performed on the integrated core to check interoperability between modules, the core was also prepared for prototyping on the FPGA for real-time verification. As individual modules were already verified for handshakes between modules, output generated from each module for a set of inputs given by the previous module verified for functional correctness.

To reduce verification overhead on prototyping performance and to improve observability, proper interface logic was inserted in the memory controller module to tap decoded frame data going to the memory. This data was transferred to the computer through USB port for monitoring and analysis. With this logic, data could be analyzed at the frame level only and did not give much insight about the source of the problem. To improve observability and controllability, the above technique was modified to monitor specific module, switch inputs were used to select the module whose data needs to be tapped. Using this technique it was possible to verify individual modules. Video sequences that did not pass the frame level tests were passed through module level tests. Data from each individual module was tapped and verified. Input data available for each erroneous module was used to simulate the module and identify the error. This helped

to reduce transfer of redundant data which delays verification process and also reduced the amount of data that needs monitoring and analysis. Module level tapping helped to achieve faster verification and avoided frame level simulations.

To automate the verification process, simple software programs were written to capture data transferred from the board and to compare it with the expected values generated from H.264 reference software [50, 51]. The data that generated errors were manually analyzed and the error source was debugged. For complex errors, simulation was performed on modules with identified error cases to analyze the error and the bugs were fixed by modifying the code. To further improve controllability and observability switches were used to control frame-by-frame decoding. This helped to keep track of frame numbers that produced errors and made verification easy. Decoded frames were displayed on the monitor to track visual artifacts and quick verification of various video sequences. The proposed technique helped to quickly verify the decoder design, almost in real-time, for different video sequences with various levels, modes, frame resolutions and bit rates. Once the functional verification of the design was complete, H.264 decoder was available for real-time system level prototyping with minimal modifications saving lots of design time.

4.3.1 Results

Charts in Figures 4-3, 4-4, 4-5 show the gains of proposed technique over the simulation and emulation based verification. Figure 4-3 shows the average verification time taken by each of the techniques to verify 50 frames of a 4CIF sequence.

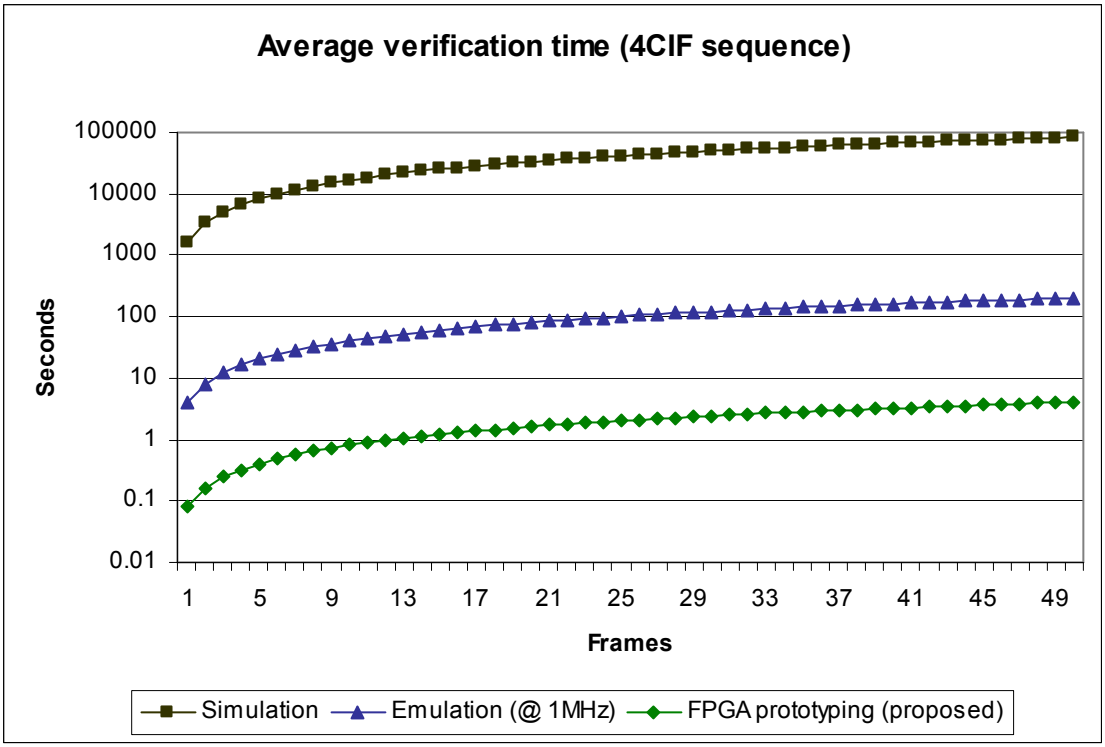

Figure 4- 3: Average verification time for 4CIF sequence

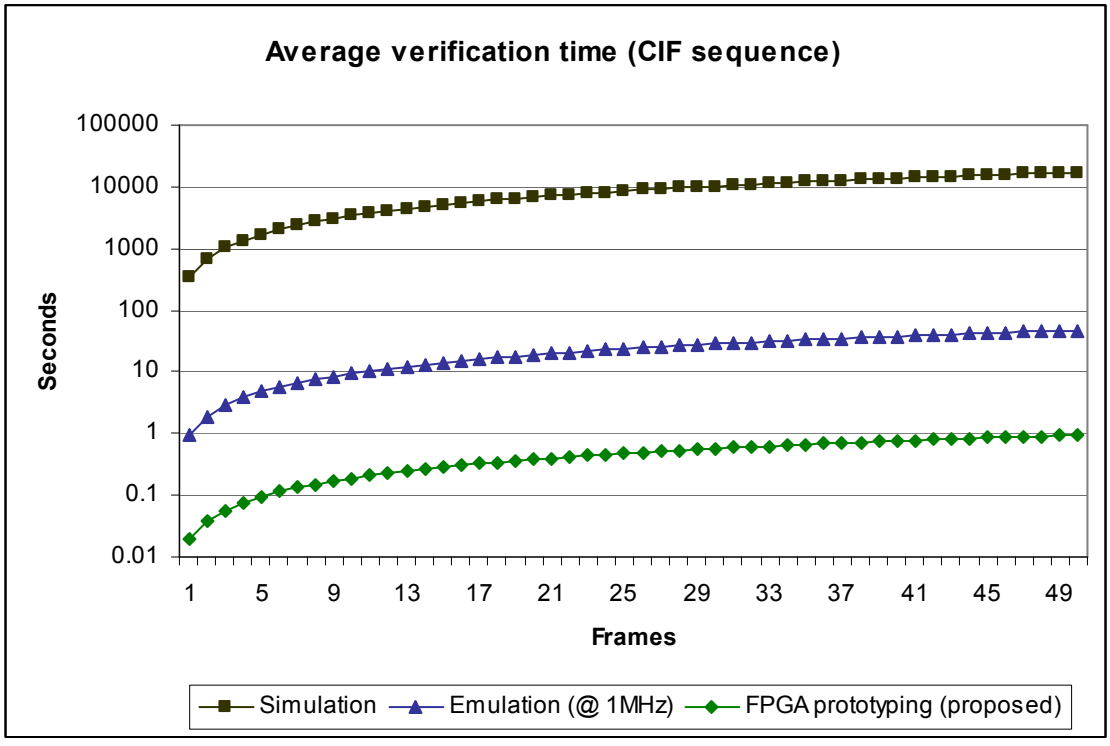

Figure 4- 4: Average verification time for CIF sequence

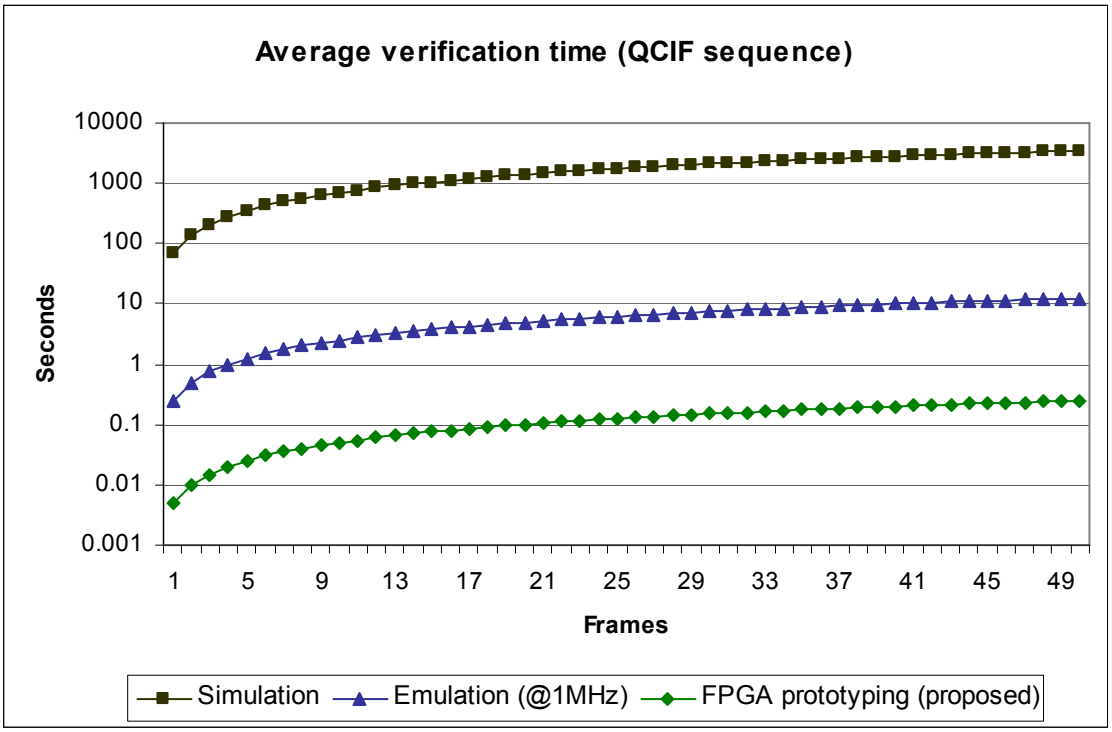

Figure 4- 5: Average verification time for QCIF sequence

Figure 4-4 and Figure 4-5 shows average verification time taken by simulation, emulation and proposed technique for CIF and QCIF sequences respectively. In all the cases we can observe that the proposed technique performs 15000 to 20000 times faster than simulation based verification and at least 50 times better than the emulation based verification.

4.3.2 Advantages and limitations

FPGA prototyping based verification gives significant advantages in terms of speed compared to simulation and emulation based verifications. The functional verification can be performed almost real-time. FPGA prototyping based verification is also a significantly low cost alternative compared to emulation based verification and

also helps to reduce the design time. Also, the decoded sequences can be visually inspected for correctness. This technique also helps to test the interface logic like memory controller and display interface at system level in real-time which is not possible through simulation.

Unmatched performance in speed and cost effectiveness of prototyping based functional verification comes with some limitations. This technique gives lesser observability compared to simulation tools. Though the end results of each module can be easily tapped and monitored, signals within the module cannot be monitored to identify the source of the error. In other words, this technique can quickly identify the problem but does not give much insight to resolve the same. In some cases simulations have to be run on identified problem areas of the design to further resolve them. Another drawback of this technique is the recompilation time. Synthesis and implementation tools take long time to recompile the modified code, typically two hours in case of H.264 decoder, which make the technique less useful in the initial stages of the design. This technique is well-suited to verify ASIC designs that perform block based data processing with limited data ports, as the output results can be easily tapped for monitoring without much design changes or modifications. The same technique becomes a design overhead or comparatively slower to verify designs that require monitoring lot of data or signals at every clock. Any problems caused due to improper handshakes between the modules are hard to identify.

4.4 Summary

In this chapter, FPGA and prototyping boards are introduced. Criteria to select the FPGA and prototyping board for prototyping and verifying an ASIC design are discussed. FPGA prototyping based functional verification is proposed and its advantages and limitations are discussed. Section 4.3 explains the verification technique employed and proves that FPGA prototyping based functional verification helps to reduce the functional verification time and the overall design time significantly and is a better alternative compared to simulation and emulation based verifications with some limitations. The following chapter summarizes the work and presents comparative results. Future research interests are discussed in brief.

CHAPTER 5

CONCLUSIONS AND FUTURE RESEARCH

In this thesis, functional verification of ASIC H.264 decoder is discussed. The limitations of existing techniques, such as simulation and emulation based verification, to verify complex designs like video codec are highlighted. FPGA prototyping based functional verification is suggested as a faster and economical alternative. In this concluding chapter, advantages of the proposed techniques are summarized and comparison with existing techniques is show. Finally, future research interests and extensions to this work are discussed.

5.1 Conclusions

Robustness and proper functioning of an engineering design can be assured only by proper testing and thorough functional verification. For ASIC H.264 decoder design, thorough functional verification is possible only by verifying the design for long video sequences with various encoding levels, modes, frame resolutions and bit-rates. With the limitations in speed of operation to simulate ASIC H.264 decoder as discussed in chapter 3, simulation technique becomes a bottle neck and consumes lot of design time. Thorough verification is not possible through simulation tool as it would take two days just to simulate 120 frames of a 4CIF sequence. Although emulation based verification is a faster alternative compared to simulation, it cannot give real-time performance.

Also emulation tools are expensive and take longer learning time. Also the compilation time after every design change is more. To overcome limitations of both the existing techniques, proposed FPGA prototyping based verification technique makes use of custom off the shelf FPGA prototyping boards that are inexpensive compared to emulation tools and perform significantly faster than both emulation and simulation based verification techniques. This technique is ideal for designs like H.264 decoder [4] which do block based data processing. Table 5-1 compares the proposed technique against existing verification techniques.

Table 5- 1: Feature comparison of proposed technique with existing techniques

SL. NO.	Feature	Simulation based verification	Emulation based verification	FPGA prototyping based verification (proposed)
1	Controllability	Excellent	Moderate	Moderate
2	Observability	Highest	Moderate	Least
3.	Speed of operation	Slowest	Moderate	Real time
4.	Cost	Inexpensive	Very expensive	Inexpensive
5.	Compilation time	Low	High	High
6.	Verification with real hardware	NO	Partial	Yes
7.	Thorough verification	Not possible	Possible	Practical
8.	Verification of interfaces	Not possible	Partially possible	Possible

Looking at the comparison (Table 5-1) we can conclude that proposed technique is a significantly faster and cost effective alternative compared to existing verification techniques. Proposed technique can save a lot of design time and project costs if properly used and adopted well within the ASIC design process. This technique can give real time performance and can also verify interface logic at hardware level. With real time performance thorough verification is possible making the design robust for ASIC fabrication.

5.2 Future research

Any new technique has scope for improvements. With the technology advancing faster than ever, newer interfaces can be used to monitor signals in the FPGA. Also, interfaces with fast data rates can be tried to transfer more data from the design for analysis and monitoring. Automated techniques can be developed to capture and monitor data. Also tools can be added to present the data to the designer in a convenient way to debug the problem. Software interfaces can be developed to interact with the design in the FPGA and to monitor the performance of the design modules.

REFERENCES

[1] MPEG-4 : ISO/IEC JTC1/SC29 14496-10: Information technology - Coding of audio-visual objects - Part 10: Advanced Video Coding, ISO/IEC, 2005

[2] H.264: International Telecommunication Union, Recommendation ITU-T H.264: Advanced Video Coding for Generic Audiovisual Services, ITU-T, 2003

[3] Synplicity Inc., "ASIC Prototyping Using Off-the-Shelf FPGA Boards", white paper, Jan 2006. http://www.techonline.com/pdf/pavillions/synplicity/synplicity_prototyping.pdf

[4] K. Kwon, A.Tamhankar and K.R.Rao, "Overview of MPEG-4 Part 10". Journal of Visual Communication and Image Representation, Vol. 17, Issue 2, pp. 186-216, Apr 2006.

[5] ITU-T website for H.264 standard, http://www.itu.int/rec/T-REC-H.264/en, Apr 2006.

[6] MPEG-2: ISO/IEC JTC1/SC29/WG11 and ITU-T, ISO/IEC 13818-2: Information Technology - Generic Coding of Moving Pictures and Associated Audio Information: Video, ISO/IEC and ITU-T, 1994

[7] MPEG-4: ISO/IEC JTCI/SC29/WG11, ISO/IEC 14 496:2000-2: Information Technology-Coding of Audio-Visual Objects-Part 2: Visual, ISO/IEC, 2000.

[8] G. Sullivan, P. Topiwala and A. Luthra, “The H.264/AVC advanced video coding standard: overview and introduction to the fidelity range extensions”. SPIE Conference on Applications of Digital Image Processing XXVII, vol. 5558, pp 53-57, 2004.

[9] I. Amer, W. Badawy, and G. Jullien, “Towards MPEG-4 part 10 system on chip: a VLSI prototype for context-based adaptive variable length coding (CAVLC)” IEEE workshop on signal processing systems, pp. 275-279, Oct 2004.

[10] H Lin et al., “Combined 2-D transform and quantization architectures for H.264 video coders”, IEEE international symposium on circuits and systems, vol. 2, pp. 1802 – 1805, May 2005.

[11] A. Puri, X. Chen and A. Luthra, “Video coding using the H.264/MPEG-4 AVC compression standard”, Signal Processing: Image Communication, vol. 19, issue 9, pp. 793-849, Oct 2004.

[12] Y. Huang et al., “Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder”, IEEE Transactions on circuits and systems for video technology, vol. 15, No. 3, pp. 378-401, Mar 2005.

[13] I.E.G.Richardson, “H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia”, John Wiley & Sons, 2003.

[14] S.-C. Chang et al., “A Platform Based Bus-interleaved Architecture for De-blocking Filter in H.264/MPEG-4 AVC”, IEEE Transactions on Consumer Electronics, Vol. 51, No. 1, pp. 249-255, Feb 2005.

[15] V. A. Pedroni, “Circuit Design with VHDL”, ISBN 0-262-16224-5, MIT press, 2004.

[16] IEEE Std 1076-1987: IEEE standard VHDL language reference manual, Mar 1988.

[17] IEEE Std 1164-1993: IEEE standard multivalue logic system for VHDL model interoperability (Std_ logic_ 1164), May 1993.

[18] IEEE Std 1076-2002 (Revision of IEEE Std 1076, 2002 Edn): IEEE standard VHDL language reference manual, 2002.

[19] M. K. Dhodhi, I. Ahmad and S. Tariq, “Functional verification of multi-million gates ASICs for designing communications networks: trends, tools and techniques”, The Eleventh International Conference on Microelectronics, pp. 97-100, Nov 1999.

[20] A. Randjic et al., “Complex ASICs verification with SystemC”, 23rd International Conference on Microelectronics, pp. 671-674, May 2002.

[21] EDA Industry Working Groups, http://www.eda.org/, Apr 2006.

[22] O. Tschäche and V. Sieh, “ATOMS - A Tool for Automatic Optimization of Gate Level VHDL Models for Simulation”, Proceedings of 8th European Simulation Symposium, vol. II, Oct 1996.

[23] A. Hoffmann, T. Kogel and H. Meyr, “A framework for fast hardware-software co-simulation”, IEEE Proceedings of Design, Automation and Test, pp. 760 – 764, Mar 2001.

[24] J. A. Rowson, “Hardware/Software Co-Simulation”, 31st ACM/IEEE Design Automation Conference, pp. 439 – 440, Jun 1994.

[25] W. I. Baker and A. Mahmood, “An analysis of parallel synchronous and conservative asynchronous logic simulation schemes”, Proceedings of sixth IEEE Symposium on Parallel and Distributed Processing, pp. 92 – 99, Oct 1994.

[26] M. L. Bailey, “A time-based model for investigating parallel logic-level simulation”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 11, issue 7, pp. 816 – 824, Jul 1992.

[27] Cadence Design systems, website for information on NC-VHDL simulator, http://www.cadence.com/products/functional_ver/nc-vhdl/index.aspx, Apr 2006.

[28] H. N. Nguyen and M. Thill, “Design verification based on hardware emulation”, Proceedings of Seventh IEEE International Workshop on Rapid System Prototyping, pp. 2 – 4, Jun 1996.

[29] S. Walters, “Reprogrammable hardware emulation for ASICs makes thorough design verification practical”, Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage, Digest of Papers, pp. 484 – 486, Mar 1989.

[30] Mentor Graphics Corporation, VStation Pro – emulation platform, http://www.mentor.com/products/fv/emulation/vstation_pro/upload/vstation_pro.pdf, Apr 2006.

[31] Cadence Design Systems Inc., Palladium series of accelerators/emulators, http://www.cadence.com/datasheets/incisive_enterprise_palladium.pdf, Apr 2006.

[32] C. Chuang et al., “A snapshot method to provide full visibility for functional debugging using FPGA”, IEEE Proceedings of 13th Asian Test Symposium, pp. 164 – 169, Nov. 2004.

[33] S. Brown and J. Rose, “FPGA and CPLD architectures: a tutorial”, IEEE Design & Test of Computers, vol. 13, issue 2, pp. 42 – 57, Summer 1996.

[34] P.-A. Hsiung, “Hardware-software timing coverification of concurrent embedded real-time systems”, IEE Proceedings on Computers and Digital Techniques, vol. 147, issue 2, pp. 83 – 92, Mar 2000.

[35] Y. Sungjoo et al., “Fast hardware-software coverification by optimistic execution of real processor”, Proceedings of IEEE Conference and Exhibition on Design, Automation and Test, pp. 663 – 668, Mar 2000.

[36] T.W. Albrecht et al., “HW/SW coverification performance estimation and benchmark for a 24 embedded RISC core design”, Proceedings of IEEE Conference on Design Automation, pp. 808 – 811, Jun 1998.

[37] Xilinx Inc., “Virtex-4 Family overview”, http://direct.xilinx.com/bvdocs/publications/ds112.pdf, Feb. 2006.

[38] Xilinx Inc., “Important Dates in Xilinx history”, http://www.xilinx.com/company/xilinxstory/timeline.htm, Apr 2006.

[39] S. Brown, “FPGA architectural research: a survey”, IEEE Design & Test of Computers, vol. 13, Issue 4, pp. 9 – 15, Winter 1996.

[40] Xilinx Inc, “Virtex-4 Family overview”, http://direct.xilinx.com/bvdocs/publications/ds112.pdf, Apr 2006.

[41] Xilinx Inc. website, http://www.xilinx.com/, Apr 2006.

[42] Xilinx Inc. white paper, “Achieving breakthrough performance in Virtex-4 FPGAs”, http://direct.xilinx.com/bvdocs/whitepapers/wp218.pdf, Apr 2006.

[43] Xilinx Inc., “Gate Count Capacity Metrics for FPGAs”, http://direct.xilinx.com/bvdocs/appnotes/xapp059.pdf, Apr 2006.

[44] Xilinx Inc. ISE foundation, tool for synthesis and implementation, http://www.xilinx.com/ise/logic_design_prod/foundation.htm, Apr 2006.

[45] The Dini Group, DN8000K10PCI Virtex4 Based ASIC Prototyping Engine, http://www.dinigroup.com/DN8000k10pci.php, Apr 2006.

[46] The Dini Group website, http://www.dinigroup.com/, Apr 2006.

[47] Intel corporation, website for information on PCI, http://www.intel.com/ standards/case/case_pci.htm, Apr 2006.

[48] USB Implementers Forum, Inc., website for information on USB, http://www.usb.org/, Apr 2006.

[49] The Dini group, Virtex-4 based Digital Video Interface (DVI) daughter-card, http://www.dinigroup.com/dvidc.php, Apr 2006.

[50] ITU-T Recommendation H.264.2: Reference software for H.264 advanced video coding, http://www.itu.int/rec/T-REC-H.264.2-200503-I/en, Mar 2005.

[51] Website for current version of H.264 reference software and software documentation, http://iphome.hhi.de/suehring/tml/, Apr 2006.

BIOGRAPHICAL INFORMATION

Basavaraj Mudigoudar received his Bachelor of Engineering degree in Electronics and communications engineering from Karnataka University, Dharwad, India in October 2000. After working in the industry for three years developing ASIC video and audio codecs like MPEG-4 and MPEG-1 part 3 (MP3), he pursued his masters studies at University of Texas at Arlington. He was the member of digital image processing research group guided by Dr. K. R. Rao. He received his M.S. degree in Electrical engineering in May 2006 from Universtiy of Texas at Arlington. He worked as intern and is also currently employed with FastVDO LLC, Columbia, MD. His research interests are multimedia processing, ASIC implementation and FPGA prototyping.