

VHDL Methodologies and Management

John S. Hong, Juergen Lutz, and Patty Sanchez
Digital Signal Processing Division
LSI Logic Corporation
1551 McCarthy Blvd., MS G-815
Phone: (408) 433-8000
Fax: (408) 954-4855
Milpitas, CA 95035

Abstract

This paper describes our experience developing the methodologies and design management needed to use VHDL as the common simulation language for the design of a DSP chipset. This chipset is the core of a second-generation CCITT H.261 videophone application. Our first requisite in the design process was to determine system and chip partitioning. From there we had to determine which internal and third-party vendor tools to use to design the chipset. A top-down design methodology approach was used on the entire chipset; but because portions of the original design were reused, we also developed various design import methodologies. Finally, in order to facilitate a smooth design flow, we paid close attention to the logistics of design management.

Introduction

The redesign of the chipset using VHDL needed special design considerations. Six chips were required to complete the chipset. Two chips from the existing design were reused; however we had to deliver VHDL models of these chips so our customer could perform system simulation. The four remaining chips were built from scratch. Some of these chips included internal blocks that had been developed prior to the beginning of this project. In order to integrate these existing blocks, a different design methodology was developed for each chip.

The design methodology associated with each chip required different software tools. The following describes the tools and languages used to implement and verify the chipset at both the chip and system level:

- **Concurrent Module Design Environment™ (C-MDE™) Design System**—LSI Logic's C-MDE design system is a workstation-based, comprehensive CAD environment that features concurrent processing; automatic design compiling, linking, and delay predicting; a consistent, easy-to-use interface; and shared memory.
- **Silicon 1076**—LSI Logic's Silicon 1076 is a VHDL-based system development environment.
- **VantageSpreadsheet®** —Vantage® Analysis System's VHDL simulation environment.
- **Synopsys Design Compiler™**—Synopsys™, Incorporated's synthesis and optimization tool.

- Logic Expression Synthesis (LES) Language—LSI Logic’s LES language is a high-level, digital hardware description language suitable for behavioral simulation, mixed-mode simulation, and automatic logic synthesis.
- Network Description Language (NDL)—LSI Logic’s proprietary gate-level design description language used in C-MDE environment.

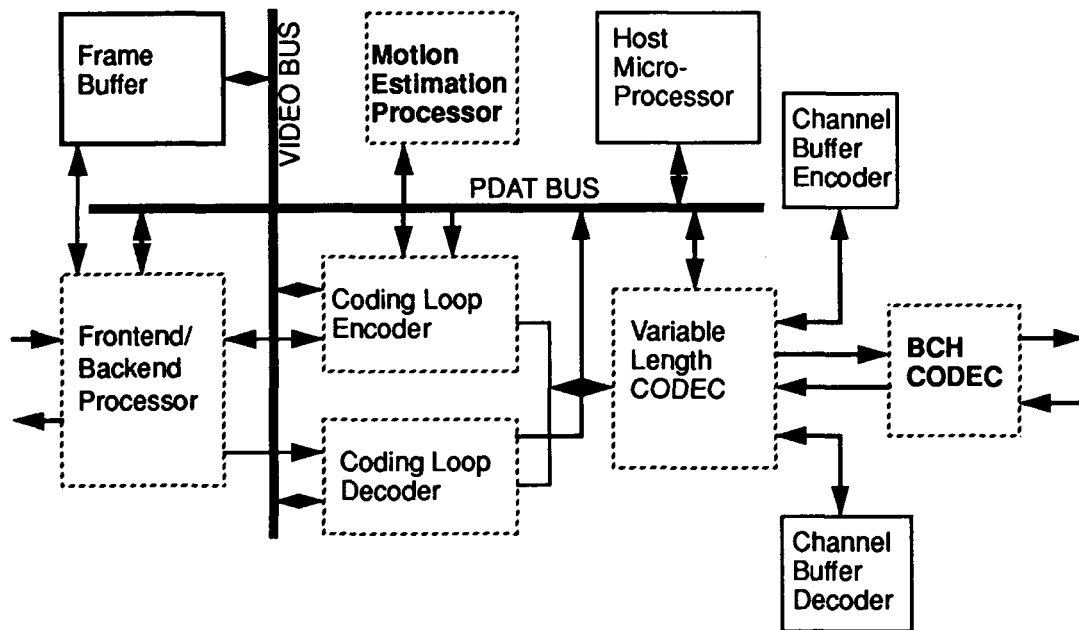
Project Goal

Our goal was to design and deliver VHDL models and silicon for the following chips:

- frontend/backend processor
- coding loop encoder
- coding loop decoder
- variable length coder/decoder
- motion estimation processor
- BCH CODEC

Figure 1 shows an overview of the entire system with the CCITT H.261 encoder/decoder chipset. The six chips of the chipset are outlined using a dotted line; the two existing chips are shown in bold font.

Figure 1. System Overview with the CCITT H.261 Encoder/Decoder Chipset



Initial Design Status

When we initially began the design process of the CCITT H.261 encoder/decoder chipset, we examined the status of each required chip. The motion estimation processor and the BCH CODEC chips were carried over from the first generation chipset and required only

the generation of VHDL behavioral models. The frontend/backend processor would be designed from scratch. The coding loop encoder and coding loop decoder would be designed from scratch but would use existing modules. The variable length coder/decoder would encompass the encoder/decoder cores from the first-generation chipset, but the chip-to-chip interface would need to be developed; fortunately, a netlist format and LES format of the two cores were available.

Next, we examined benefits of top-down design methodology.

Top-Down Design Methodology

We kept the following three factors in mind as we used the top-down design methodology approach: design flow, implementation, and verification.

The design flow starts with system-level partitioning and ends with simulation. We were required to partition the chipset using the specifications from the *CCITT Video CODEC for Audiovisual Services at $p \times 64$ kbits/s, Recommendation H.261*. We also took into account gate count, RAM usage, and the reusability of chips from the first-generation chipset. Once the design was partitioned into functionality, we had to determine the best way to simulate. System simulation requires behavioral and RTL-level code; gate-level code should be avoided to maximize speed of simulation. Chip simulation requires RTL and gate-level code.

Implementation of the top-down methodology involves synthesis and optimization. The tool we used to synthesize varied depending on the type of module. For random logic or state machines we used Synopsys' Design Compiler synthesizer. Memories were synthesized using LSI Logic's Memory Compiler program, which is part of the C-MDE design system. For logic blocks (such as adders, multipliers, or counters), we used LSI Logic's Block Synthesis program, which is also part of the C-MDE design system. Logic functions can be synthesized by the Design Compiler synthesizer; however we chose to use Block Synthesis because Block Synthesis provides both synthesis and layout optimization. Those elements synthesized by the Design Compiler synthesizer must also be optimized by the Design Compiler optimizer. But because this optimizer did not provide layout constraints, it was necessary to change the code manually to input a different drive strength or random logic to optimize layout. LES-generated netlists were also optimized using Synopsys tools.

Finally, verification requires functionality comparison, timing verification, and testability. Throughout the simulation process, functional comparison between the different levels of code in the same block had to occur. Therefore to ensure accurate comparisons, it was important to consistently use the same I/O port types (for example, `std_logic`, integer, bit) and apply the same vector set across the different levels of code. Timing verification could be done only at the gate level using C-MDE design system. Unfortunately, C-MDE design system cannot backannotate layout information (wire length) to VHDL gate-level models. In order to maintain consistency across code levels, we had to make timing changes at the RTL level and resynthesize. We then used the JTAG (Joint Test Action Group) test standard to test each chip.

Top-down methodology does not address the problem of importing existing chips and functional blocks into the design; a design import methodology was needed.

Design Import Methodology

The structure of the chip or functional block determined which design import methodology to use. If the chip contained only glue logic and had an LES-generated netlist, we used LSI Logic's LTB program, which translates the LES netlist to VHDL and produces a

functional model of the circuit. If the chip's netlist was generated using LSI Logic's proprietary Network Description Language (NDL), we used Silicon 1076 to translate the NDL netlist to VHDL. A chip that contains memory must be imported using a different methodology. One method required partitioning the memory block at the top level; describing the memory block using register descriptions, which makes the RAM block appear as glue logic; and using LES to translate the entire chip. Another method available required partitioning the memory, describing the RAM at the behavioral level using VHDL, using LES to translate the remaining glue logic, and then connecting the RAM and the glue logic together.

How a functional block was imported depended on whether it was a Block Synthesis-generated module (for example—adders, multipliers, FIFOs, etc.) or whether it was a memory module. To handle non-memory module importation, we wrote behavioral-level code using the same I/O pins as the Block Synthesis module and then used instantiation to replace the module. Silicon 1076 automatically creates VHDL models of memories. If, however, the memory was generated in a library not supported by Silicon 1076, we had to describe the module at the behavioral level and then instantiate the memory block. Since the behavioral-level code did not include exact timing, timing had to be verified during gate-level simulation.

Actual Implementation

This section describes how we implemented both the top-down and design import methodologies to design the chips for the CCITT H.261 chipset.

- Frontend/backend processor

Based on the necessary functionality this chip required, it had to be designed from scratch and could not make use of any existing modules. We decided to describe the entire chip as a RTL model for ease of simulation and job share. Several designers worked concurrently on the functional blocks that made up this chip and then synthesized the functional blocks into one netlist.

- Coding loop encoder/Coding loop decoder

These chips incorporated modules that were prepared before the project began. We decided to describe each chip as a behavioral model because we did not need to synthesize all the modules, our customer required VHDL models as early as possible, and behavioral models simulate in the system faster.

- Variable length CODEC

This chip encompassed the variable length encoder chip and the variable length decoder chip from the first-generation chipset. These original chips were done in LES. We decided to use LES to describe the new chip and then translate it to VHDL. The design engineer's expertise with LES made it possible to easily convert two chips into the single chip; VHDL training would have increased the design cycle.

- Motion estimation processor and BCH CODEC

These chips already existed in silicon and required only the generation of behavioral models.

Design Management

The following describes the steps we took to manage the CCITT H.261 design process:

- Step 1. We partitioned the system according to the functional blocks specified in the H.261 specification released by the CCITT and according to the optimal chip size. We then partitioned the chips according to functional blocks and memories.
- Step 2. We adopted design methodologies that avoided design redundancies. For example, a set of VHDL packages—which contained subtype definitions, common functions, etc.—were created and shared by the designers. In addition, designers reported problems and how the problems were resolved so the group's design methodology could be altered accordingly.
- Step 3. Design engineers were assigned projects based on their expertise. Table 1 outlines each design engineer's experience with VHDL, familiarity with the overall function of the chipset, and the assignments.

Table 1. Evaluation of Design Engineer's Experience

Designer	VHDL Experience	Familiar with Function of Chipset	Chip/System Assignment
A	Good	Yes	FBP ¹ , System
B	None	Yes	VLC ² , System
C	Better	Yes	CLE ³ , CLD ⁴ , FBP
D	Good	Yes	FBP
E	Better	Minimally	FBP
F	Good	Minimally	CLD
G	None	Minimally	CLE

1. Frontend/Backend Processor
2. Variable Length Coder/decoder
3. Coding Loop Encoder
4. Coding Loop Decoder

- Step 4. We took advantage of the VHDL design environment by performing parallel design and verification processes. For example:
 - Multiple designers wrote VHDL code for different functional blocks contained in one chip; I/O ports were clearly defined in advance to avoid errors.
 - Experienced designers concentrated on writing the RTL code and then passed the code to less experienced VHDL users who synthesized the code and compared the results with the previous levels of code.
 - System simulation and chip implementation were done at the same time. This parallel approach allowed our customer to validate the behavioral chip models without delaying silicon.

Conclusion

The advantages of using VHDL to redesign the chipset were numerous: software language barriers between developers were removed, adequate documentation of the chips was generated, and the interface between multiple VHDL-related design tools was seamless. The success of this design proves that VHDL can be used at any level of the design process (behavioral system simulation, chip simulation, and chip implementation), and VHDL can be used to fully verify the design and the system.