VLSI Design and System Level Verification for the Mini-Disc

Tetsuya Fujimoto , Takashi Kambe

SHARP Corporation
Nara 632 JAPAN

Abstract — In this paper, a new method for the design of complex multimedia ASIC is introduced. Using this design method, VLSI with embedded software and high-speed emulators can be developed concurrently. The method has proven to be effective through actual design of VLSI for audio compression and decompression in a Mini-Disc system.

1 Introduction

To be competitive in the consumer electronics market, it is necessary to enhance both cost and performance of products rapidly, during their very short lifetime. A typical case is Mini-Disc(MD), a new digital audio system, in which we have experienced such a scenario in the last four years. MD has the following two great advantages over CD(Compact Disc). In MD, an audio compression and decompression technology is adopted. Playing time per disc, 75 minutes, is the same as CD but MD needs only less than one-fourth of the disc volume. Another feature is that the system is recordable, which means that MD can take the place of conventional cassette tape. For this reason MD is a very promising product in the consumer electronics market.

We have developed several generations of audio compression and decompression LSI for the MD system over the last four years. The most important requirement is, of course, sound quality because this is an audio system. Needless to say, short development time and low chip cost are very important, but we also have to consider the market requirements. In the early stage of market growth, the VLSI design time must be short because shipping new product earlier than competitors brings larger market share and a leading market position. On the other hand, cost must be lowered to be more competitive when the market becomes saturated.

A top-down VLSI design methodology is considered to be a solution. It is well known that we can achieve high design productivity and short development time by designing with an accurate specification for an LSI and design verification. It is therefore a matter of course that we adopted a top-down design methodology.

An open problem is how to guarantee the consistency between a specification for an LSI and requirements of a product. It is difficult to define the former especially when the latter is complex or is not very deterministic. In the case of a complex system such as a multi-media application it is very difficult to provide an accurate specification for an LSI early in the design cycle. In another case, for example, an LSI tightly coupled to analog or mechanical components, we sometimes cannot determine the specification at all without first making the characteristics of the other components clear. Eventually an LSI may fail to work correctly in the target system for this reason.

Our LSI is also a typical case. The decompression algorithm is the standard. However many other factors affect to the sound quality. These factors are compression algorithm, error concealment method, round-off error of fixed point arithmetic and so forth. Everything is integrated in an LSI and then embedded into the MD system. The sound quality must be good enough in the product. Hence, it is difficult to define and verify the specification. So far, the CAD technologies which are available have not provided a solution to this problem.

We can develop a correct LSI, using a top-down design method, only when the specification is correct. Thus we need some method of performing system level verification, which is mainly the verification of an LSI's specification against the requirements of a system.

In the case of our LSI, the final requirement is sound quality and this can only be judged by listening to the actual sound. No simulation tool is capable of doing this. HDL simulation is too slow to be used for real sound, and simulation using C-language software is too far from the LSI implementation.

We decided to develop a high-speed emulator to listen to and evaluate the actual sound generated by an LSI. Along with the development of LSIs, we have established a new design method for developing LSIs with complex specifications, and concurrently developing an emulator for verification of the LSI at the system level.

This paper is structured as follows: The MD system is briefly described in section 2. A design environment is described in section 3. The method for the concurrent design of both an LSI and its emulator, one of the most important aspects of our method, is presented in section 4. Finally, the results of appling this method to the

33rd Design Automation Conference ®

audio compression and decompression LSI are shown in section 5.

2 MD system and Audio compression and decompression algorithm

The MD system resembles the CD(Compact Disc) system very closely, as shown in Fig.1. According to the advantages of MD over CD as explained in the section 1, we note the following major differences:

– In play mode, after signal processing in common with CD (EFM demodulation and CIRC error correction), compressed audio data, not an audio signal, is obtained. A decompression operation is then necessary to obtain an audio signal from this data.

– The signal processing system is bi-directional.

These differences are shown in Fig.1 by a shaded area. In this paper, we will mainly discuss the design method for the compression and decompression LSI, indicated by "COMPAND" in Fig.1. To design the chip, the compression and decompression algorithm "ATRAC" (Adaptive TRansform Acoustic Coding) used in the MD system, must be understood.

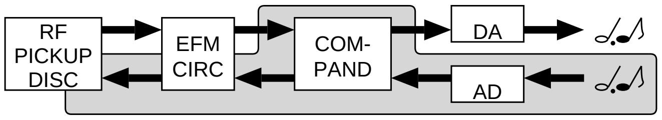

Figure 1: MD system

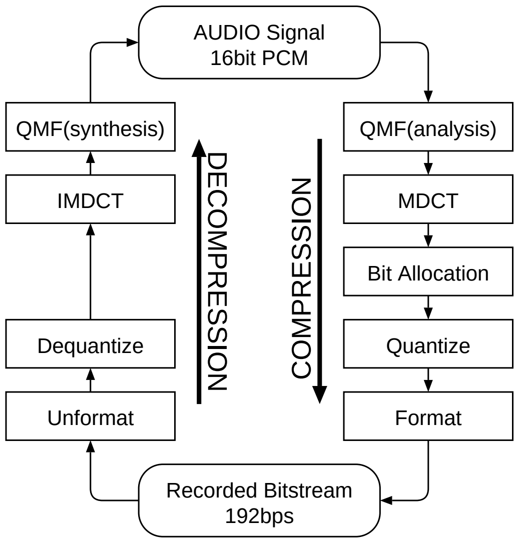

Figure 2: ATRAC algorithm

The ATRAC compression algorithm is outlined in the right half of Fig.2. The input is a digital audio signal, which is 16bit PCM code at 44.1 KHz sampling frequency. First, in the signal processing steps, the input signal is analyzed by QMF(Quadrature Mirror Filter) into three frequency bands. An MDCT(Modified Discrete Cosine Transform) is then applied and each frequency band is transformed from the time domain into the frequency domain to obtain the audio spectrum. In the next compression step, bit allocation is calculated to quantize the audio spectrum. Finally, the quantized audio spectrum is formated into a bit stream with bit allocation and quantization information. The compression ratio is 1/4.83.

Decompression is basically the reverse process of compression, as shown in the left half of Fig.2. First, the input bit-stream(294Kbps) is unformated to an audio spectrum. The inverse transform of compression follows, that is inverse quantization, inverse MDCT(IMDCT), and QMF. The output is an audio signal, again 16bit PCM code at 44.1 KHz sampling frequency.

Although the ATRAC algorithm is complex, a few transforms dominate the total calculation time. These are the multiplication and add in QMF, the butterfly operation in MDCT, the min-max operation in the bit allocation step, and so forth. Consequently our goals were:

– To design hardware(LSI circuits) by means of the top-down design method

– To develop ATRAC compression and decompression software on the hardware designed above.

– To develop an high-speed emulator to verify the specification of the designed LSI with the requirements, especially in terms of sound quality.

3 Design Flow and the Design Environment

In order to support the concurrent development of LSI hardware, embedded software, and an emulator, a new design method is necessary. We have extended the conventional top-down design approach to include software and emulator development support as shown in Fig.3. The new design environment is organized so that all three of the design and development goals can be achieved.

3.1 Hardware/Software Concurrent Design

The LSI is dedicated to the ATRAC algorithm, and its architecture and software are tightly coupled.

We have introduced GAIO Technology Inc.'s XASS system[3] to support this design stage. The system is a set of general purpose cross-platform software development tools. In our method, we have used these development tools for the assembler and the simulation debugger. The assembler development tool generates a simple mnemonic to binary converter from a definition of the mnemonics, instruction code, and syntax. The debugger development tool generates a symbolic simulation debugger based on the information about the architecture which is described in a special language. The basic idea of this tool is similar to gcc.

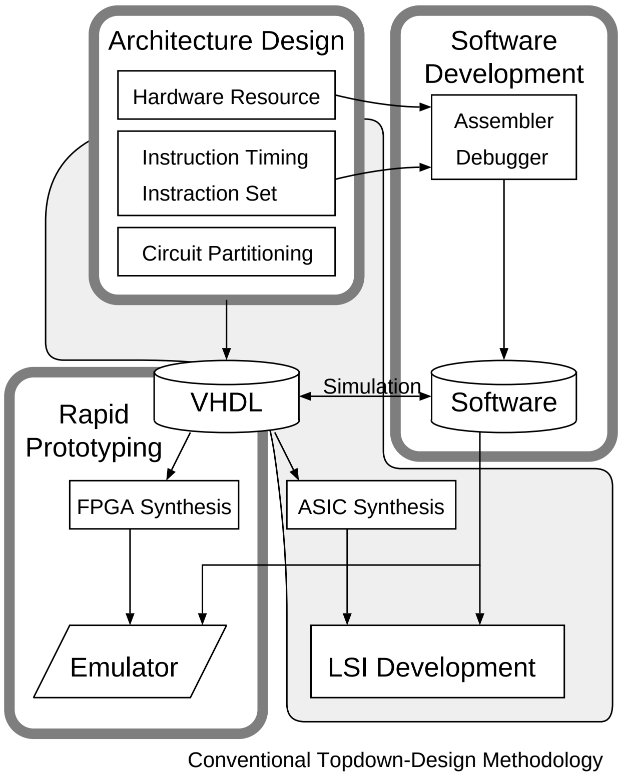

Figure 3: Design Flow

These tools have the following role in our design environment.

Assembler development tool

The architectural design goes like this:

1. Define hardware resources, block diagram, and instruction execution timing.
2. Develop an assembler using the assembler development tool.
3. Develop the fundamental routines which dominate the total ATRAC calculation time.
4. Verify the architecture in terms of the throughput of routines developed above, by the use of an ordinary VHDL simulator.

These four steps are repeated until we realize an architecture with enough throughput and with reasonable cost.

Debugger development tool

After the architectural design has been completed, a simulation debugger, which is a software model of the architecture, is developed. With the assembler and the debugger, an application software development system is then established, in which the whole specification of the ATRAC is implemented.

The simulation debugger provides debugging facilities for the target architecture with clock cycle timing. The debugger shortens software development and optimization time, and the software quality becomes good enough for physical implementation. This feature is also effective in maintaining the behavioral consistency of the hardware described by the VHDL and the debugger.

Another advantage is that the application program can be executed on the debugger one hundred times faster than on an ordinary VHDL simulator. In our case, audio data of nearly 2 seconds is computed in 1 hour of debugger time on an SS10.

3.2 LSI/Emulator Concurrent Design

There are two conflicting requirements for the design and development of an emulator. One is operation speed and the other is cost, including development time. We have used MP3, the FPCB (Field Programmable Circuit Board) system from Aptix Corp.[1] as an emulator development tool. This is a reconfigurable bread board development system which uses FPGAs and FPICs (Field Programmable InterConnect). With the MP3, Xilinx 4000 series FPGAs[2] are used to implement random logic.

Although the interconnect delay through an FPIC is significant leading to reduced emulator clock speeds, the advantage of FPICs in reducing the configuration time and effort for the emulator outweighs this.

Hence, in terms of operation speed and design cost, this is a intermediate solution between conventional bread boarding and general purpose rapid-prototyping tools such as RPM from QuickTurn Systems Inc. But we have tried to achieve as high operation speed as the former with as small development cost as the latter resolving the speed-cost dichotomy above, in the design environment in Fig.3.

4 Single design for an LSI and an emulator

The goal and the concept of single design is to use the same HDL description source for both implementation and emulation of an LSI. In particular, the architecture must be the same even if two different HDL sources are necessary for some small part of the design. This is essential in a design process that handles a large and complex design in order to avoid development delay due to the additional design effort required for the emulation, and for efforts to maintain consistency between two different designs. Our basic goal is to improve both the cost-performance of an LSI and the speed of the emulator simultaneously with the architectural design.

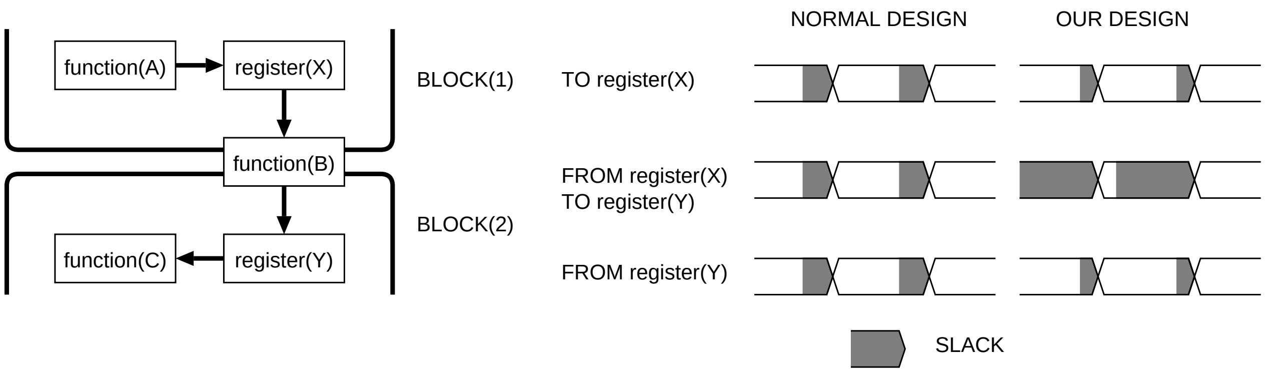

Figure 4: circuit and timing partitioning

Generally, the speed of the emulator is slower than that of an LSI based on the same architecture. Therefore it is a very important task during architectural design to make the speed of the emulator as high as possible, while at the same time keeping detrimental side effects to the cost-performance of the LSI as small as possible.

The speed degradation of an emulator comes from the following three factors:

Factor 1 On an emulator, signal propagation delay between electrical components such as FPGAs and memory chips are much larger than those in an LSI, because of the longer propagation distances, the larger wiring capacitance and input/output buffers between chips.

Factor 2 In the case of an emulator which uses programmable connection chips like the MP3 system, more signal propagation time is needed because of the delay through the connection chips.

Factor 3 A circuit implemented in FPGA is slower than one implemented in LSI. This is because signals inside the FPGA propagate through many pass-transistor switches.

4.1 Partitioning

This subsection treats factors 1 and 2 described above. At the architectural design we solve these problems mainly by means of circuit and timing partitioning, taking into account the characteristics of the emulator development technology.

In general, signal propagation delays between registers are constrained by the given clock cycle. Hence, functions are implemented between registers making full use of the given clock cycle time to derive high performance.

In Fig.4, let X and Y be registers and let A, B and C be functions. In the case of ordinary ASIC design timing constraints for each register to register transfer are the same if all registers are controlled by a single clock .

Therefore functions A, B and C are designed so that each of these delays satisfies the given constraint. At the same time, the timing margin is explored by making the maximum delay to be as small as possible. The middle portion of Fig.4 shows such a design.

In our method, no function is allowed on a register transfer path across a boundary between blocks. A block may be a function block described by VHDL code, in other words, a block may consists of only one VHDL entity. If a block is too small to be one layout block on an LSI, a merging strategy, which will be realized at the floor planning, has to be formulated in order to avoid too many block boundaries. Eventually most of the clock cycle can be considered as timing slack for signals across the block boundaries. The rightmost portion of Fig.4 shows an example. Function B may belong to the preceding pipeline stage with function A or to the succeeding stage with C, or may be divided into two parts to belong to both stages. In this design method, the timing margin may appear to degrade because the maximum delay may increases inside a block. However, the timing margin will be restored again after executing chip layout. Introducing a new pipeline stage, the partitioning stage, is also a solution if function B has large delay to distribute it to other stages.

This design strategy has the following advantages for implementing both the LSI and the emulator:

1. If a large timing margin is reserved between blocks then:
   - In the case of an LSI, the design reiterations caused by post-layout timing violation can be avoided.
   - In the case of an emulator, the maximum speed increases by using large timing slack for interconnection between components.
2. If glitches are restricted to be inside blocks then:
   - LSI's power consumption decreases because the amount of switching activity decreases on long interconnections.
   - The emulator is more stable because the amount of switching outside the FPGAs decreases.

5 Results

5.1 The effect of the method

The design flow and method has been established along with the development of second and third generation ATRAC compression and decompression LSIs.

Using the method presented, we have succeeded in developing LSIs which satisfy the given specifications with great improvement in terms of cost and power consumption, during a design cycle of one year. This is mainly because we saved much time on verification and implementation and spent more time on architectural design and software optimization.

Table 1 shows the dimensions of the 3 LSIs which have been developed. In the table, the minimum operation speed means the lower bound of clock speed required to decompress the audio signal in real time. This shows the throughput of the processor because the amount of computation for decompression is exactly determined by the algorithm of ATRAC. The comparison of power consumption is also given in play mode.

We note the following:

– In terms of function and cost, the first generation LSI supports only decompression but the second generation LSI also supports compression, which requires more speed, more software and more memory. In spite of those factors, the chip cost decreased owing to the more sophisticated architecture. In the third LSI, performance was almost doubled(++) without increasing cost.

– Cost, performance and power consumption has greatly improved and contributed to the downsizing of the portable MD system.

Generation	1	2	3
Die size(mm^2)	170	100	80
Technology(μ)	0.8	0.8	0.6
Decompression	+	+	+
Compression	−	+	++
Miscellaneous	−	−	+
Performance comparison			
Max. Operation speed(MHz)	17	24	24
Min. operation speed(MHz)	15	15	6
Power consumption(mW)	450	150	90

Table 1: ATRAC Compression/Decompression LSIs

5.2 VLSI and Emulator

We discuss the design of the third generation device described in table 1. We have developed an LSI and its emulator using the same architecture. In terms of VHDL source code, 93% of 5,066 lines for an LSI is also used by the emulator. Only on a 24 x 24 multiplier, we need 209 lines for an FPGA, instead of 334 lines for LSI. The dimensions of the LSI and the emulator are shown in table 2.

	LSI	Emulator
MP3	—	2 MP3 boards
Random logic	16K gates	XC4010 × 6
Multiplier	12K gates	XC4013 × 2
Datapath	7K gates	XC4013 × 2
RAM	66Kbits	16 SRAMs
ROM	500Kbits	13 EPROMs
Operation speed	16MHz-24MHz	10MHz

Table 2: LSI and Emulator

In the previous section we emphasized the importance of circuit partitioning for successfully achieving a single design for both LSI and emulator. In the case of the actual design, the speed of the LSI satisfied the design goal, and using the emulator we succeeded in recording and playing an audio signal in real time.

Fig.5 shows the relationship between the LSI layout and the structure of the emulator. The LSI is implemented using a building block and standard-cell method in 0.6 μ CMOS technology. The shaded blocks in Fig.5 are memories. Some standard cell blocks correspond exactly to function blocks of the VHDL code, however, others are composed of several function blocks which are too small to be placed as a whole layout block.

No VHDL code is partitioned into more than two layout blocks. The blocks are implemented on 10 FPGAs represented by numbers 0 to 9 in Fig.5. Except for FPGAs 0 to 3, each FPGA consists of several function blocks in order to use the FPGA more efficiently, to reduce number of FPGA I/O pins, and to reduce number of connections between FPGAs.

Because of the different merging criteria for LSI and emulator implementation the design hierarchy for each implementation is fixed independently. Nevertheless, the placement of function blocks on the LSI is very similar to the partitioning for the FPGAs. Each FPGA (except for FPGA9) includes function blocks which are adjacent to each other on the LSI layout.

The result indicates that the initial partitioning at the architectural design is suitable for both implementations in the LSI and the FPGAs.

5.3 Trade-offs of the methodology

However, in this development there are some small inefficiencies in terms of cost and power consumption of the LSI. This is because the concept of ensuring a single design has first priority, even higher than cost-performance.

For instance, the LSI implements only one power save mode, that is standby mode. We estimated that 3-5% of power consumption can be saved by using individual clock distribution for each block. However, the complex clock control would make it difficult to develop an emulator.

Some pipeline registers for the partitioning stage, introduced in section 4, are also redundant. We would

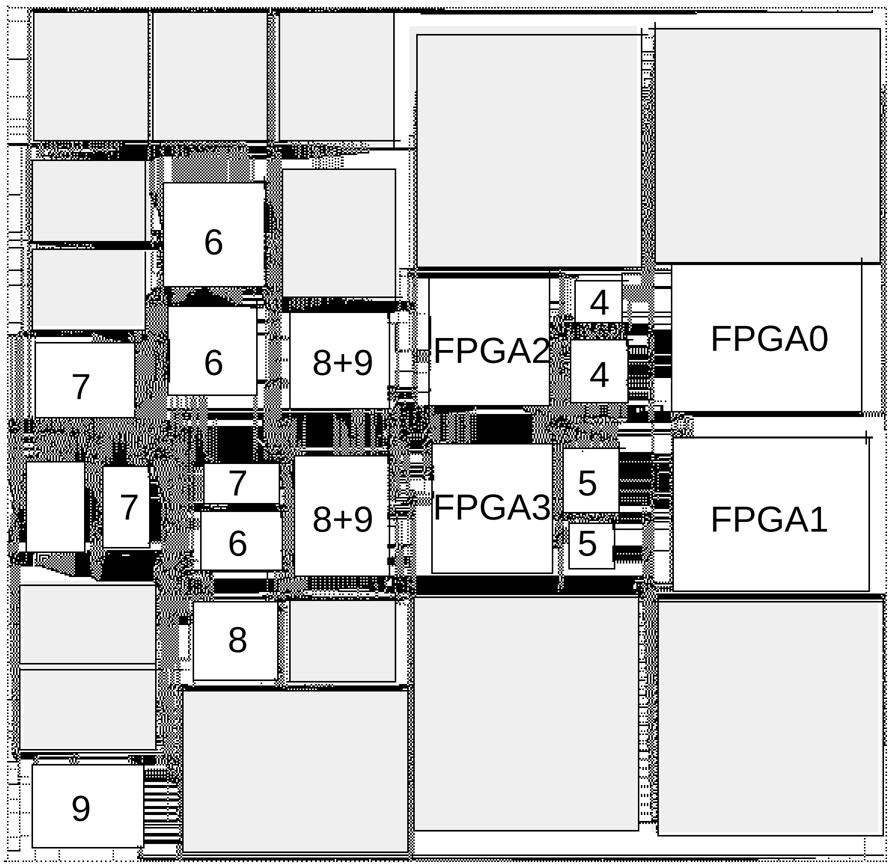

Figure 5: layout and FPGA mapping

save nearly 2,000 gates for such registers, 5.7% of the entire gate count if we were to remove them. But the random logic part is only 40% of the core area and the basic cell area is even smaller, so this would affect less than 2% of the total chip cost. Moreover, removing the partitioning stage may affect the preceding or succeeding stages and we would need more gates to satisfy timing constraints. Consequently we have concluded that the overhead is small enough in comparison with the risk of insufficient verification quality.

6 Conclusion and Future Work

An extended top-down design method is presented and applied to the design of audio compression and decompression LSIs for a Mini-Disc system. In this design environment, VLSI devices, embedded software, and high-speed emulator have been developed concurrently. We have succeeded in greatly enhancing the cost performance of the LSI and obtained a competitive key device for our product.

One important role of this method is to verify the specification of an LSI against the requirements of a system. Consequently, the top-down design method presented have become more efficient and reliable because we can avoid problems caused by the inconsistency between these two objectives.

The implementation part of the method, such as circuit design, is very well supported by various CAD technology and tools, not only for LSI but also for software and emulation. However, the early stages of development, such as architectural design, depends highly on the skill of the designer. For these design stages, several new CAD concepts and technologies are beginning to appear. In order to make the method more efficient and capable we will attempt to apply new technologies such as hardware/software co-design tools and tools for the evaluation of cost-performance and power consumption to the method in the near future.

References

[1] Aptix Corporation, “MP3 System Explorer Data Sheet” December 1994.

[2] Xilinx, Inc., “XACT Libraries Guide” April 1994.

[3] GAIO Technology Co., LTD., “General-Purpose Cross Software XASS-V series” July 1991.