Rapid Prototyping for DSP Circuits using High Level Design Tools

Stefan Tamme
SICAN GmbH, Hannover/Germany
Email: tamme@sican.de

Abstract:

Rapid prototyping allows to check digital designs working in near real time before manufacturing. This paper describes the prototyping flow for a digital audio processing circuit. This circuit has been developed and simulated with a DSP design tool. VHDL Code has been generated and synthesized. The resulting gate level netlist of the circuit has been mapped into a hardware emulator. The results of the different verification approaches are then compared in terms of setup and runtimes.

1. Introduction

Today, short time to market is one of the key factors to develop successful products. Therefore it is important that circuits work correctly without any redesigns. With modern verification tools this goal seems to be achievable, even if those are not very sophisticated. To enable a smooth interface through the design flow stages, manual interventions are necessary.

There are two major issues that make verification inefficient or incomplete:
1. The design needs a long simulation time
2. The specification is not correct

A means to overcome these problems is to emulate the design in addition to simulation. This approach is especially helpful if the design has to process a lot of data as required in digital signal processing applications. Because the emulation prototype runs at near real time it can process as much data as the final circuit. Therefore enabling verification of the correct function of the circuit by just looking at the produced output data, e.g. pictures or sounds.

If failures occur it is simple to modify the high level description and resynthesize the circuit. This avoids the problem of long simulation run times to calculate the same amount of data.

The other possible problem is an incorrect, i.e. incomplete or faulty specification of the circuits operating environment. Once an emulation model of the circuit exists, it is possible to adapt this model to the target system and to run it within its supposed environment.

Nowadays, there are a number of different tools that have to be used to transfer the design from one level of abstraction to another. In this paper, a flow using commercially available tools will be presented for these purposes. The complete design flow will be described using an echo generator for digital audio data as case study. Focus is put on the use of the prototyping system.

2. Overview of the Echo Generator

As an example to demonstrate the rapid prototyping environment a digital audio processing circuit is used, i.e. an echo generator [1]. The digitized audio data is supplied in a 32 bit wide format were the left channel sample is multiplexed in the lower 16 bit and the right channel in the upper 16 bit.

The circuit uses one of the two channels. In the next step the 16 bit are reduced down to 8 bit to limit the complexity of the circuit.

As shown in Fig. 1 a FIFO register follows in which data is delayed for a total of 768 clock cycles. In the FIFO
are three taps where the data is fed back to the input. The first of the taps appears after 576 registers. The next two follow in a distance of 96 registers each. A weighting is executed for the data that is taken out of the taps.

The first 576 registers generate a basic delay of 13 ms at a sampling rate of 48 kHz. The taps simulate obstacles that reflect the sound and that are located in different distances of the signal source.

Fig. 2 shows the output of the hall generator if it is stimulated with a sine wave input signal. During the first milliseconds the output equals zero until the first tap is reached.

![DMPNR Output](image)

**Fig. 2.** Hall generator output for sine wave input

### 3. Rapid Prototyping Design Flow

The steps that are necessary to transform the design from the algorithmic level to the gate level are illustrated in Fig. 3.

First a description of the design is entered using the DSP block libraries of Cossap. VHDL Code can then be generated based on the block diagram.

To this code some logic must be added to control the data transfer of the system. A VHDL simulator is used to verify the correct function of the whole design.

The next step is the synthesis of the design with the Synopsys design compiler. A gatelevel netlist is created by this tool and will be read into the partitioning software of the prototyping system.

The purpose of the partitioning software is to split the netlist into pieces that can be implemented within single FPGA devices.

After partitioning, the different parts of the netlist are allocated to the physically available devices of the emulator and then compiled into the programming bitstreams for the FPGAs.

The final step is the download of these bitstreams into the emulator and the verification of its correct function.

![Design Flow Diagram](image)

**Fig. 3.** Design Flow

The prototype of the chip is now available and can either be tested standalone or within the target system.

#### 3.1. Design Entry

The high level digital signal processing design tool Cossap [2] has been used as design entry tool. This tool allows the description of the application on a block level. Libraries that contain the most commonly used function blocks for DSP are supported. If the required blocks are not contained in the delivered libraries they can be generated, based on C language models.

The description can be simulated to analyze the behavior of the system. A testbench to generate and record signals can be wrapped around the design. The waveform shown in Fig. 2 is an example output diagram of Cossap.

In the audio application described here, custom-made blocks have been used to read in data that has been recorded by a Sun workstation. The Sun loudspeaker has been used for sound output.

The extraction of VHDL code is only possible if a special library is used for the block diagram.

This library is called Bittrue and contains models of blocks like adders, multipliers, registers, etc. All these blocks are scalable in their width and other parameters and can be fully simulated in the Cossap environment.

A separate tool called VHDL code generator is then used to write out the VHDL code. The VHDL code generated by this tool will be synthezizable.
3.2. Simulation

To control the interface circuitry some additional VHDL code is required. This code is added in form of a shell in which the Cossap output is instantiated as a component.

To verify the correct overall function of the design, a testbench similar to the one in Cossap is used. Data is read in and written out in ASCII format using VHDL textio functions and then converted into a format that can be played by the Sun audio tools. Thereby an acoustic check is also possible on the VHDL level.

The Vantage VSS simulator [3] has been used to perform the VHDL level simulations.

3.3. Synthesis

The whole VHDL design description is read into the Synopsys [4] environment by the VHDL compiler.

Due to the availability of Bittrue models the FIFO register is described as real register instead of RAM. The size of the register is 768 x 8 bit. In order to get acceptable synthesis times, the register has been split into 8 x 8 bit blocks that are instantiated multiple times. This splitting has been executed within Cossap.

The only constraint required to synthesize this particular design was the specification of the clock signal. There are no other constraints required because the netlist of the design has to be remapped into FPGA primitives before emulation. Therefore the timing of the whole design will differ to the emulation prototype.

To read the design into the partitioning software of the prototyping system, the netlist technology and format must be supported. The used software allows a big variety of libraries and formats. In this case the Fujitsu library CG31 has been used as target technology for the design compiler [5] and the netlist has been written out in EDIF 200 format.

The size of the produced netlist, reported by Synopsys, is 50,400 equivalent gates. Most of the gates are caused by the register chain of the FIFO, i.e. 43,000. The remaining 7,400 equivalent gates are required for the combinatorial parts of the design.

4. Emulation

A hardware emulator -also referred to as prototyping system- [6], [7], [8] is a collection of reprogrammable logic devices [9] i.e. Field Programmable Gate Arrays (FPGAs).

These devices are connected together by either special routing chips or other FPGAs that are used as routing devices, or the interconnections are simply hardwired. An illustration of an generic hardware architecture is shown in Fig. 4. In today's products, Xilinx devices [10] are most commonly used as devices to implement the logic.

Fig. 4. Emulation hardware structure

An important part of a prototyping system is the software that is required for netlist reading, partitioning, remapping and downloading to the hardware. Furthermore, debugging tools are necessary to be able to fix problems if any occur.

4.1. The prototyping hardware

The Zycad Paradigm RP 2000 (formerly Inca VA-II) system [8], used for the experiments, is able to handle up to 30 KGate designs on a single board. The board architecture is shown in Fig. 5. One board consists of eight daughtercards, each of which carrying two Xilinx 4010 FPGAs. In total there are 16 Xilinx 4010 used to map the logic. Routing is also done by FPGAs. In this case 18 Xilinx 3090 devices are used as routing chips. 32 of the IO pins of each logic device are routed to edge connectors where they can be connected to the outside world.

Fig. 5. Board architecture

An interesting feature of the system is the daughtercard architecture, that allows the combination of different daughtercards on a single motherboard. By doing this different FPGA families, memory modules or processor cores can be combined depending on the needs of the design to be prototyped.
A PC is used as host computer for download, vector apply and capture. The other parts of the software are running on a workstation.

4.2. Netlist translation

The first step is to read the design into the partitioning software. The intention of prototyping is to verify ASIC designs. Therefore it is necessary that the emulation model of the circuit is as similar to the actual ASIC implementation as possible. To achieve this, the software must be able to read the ASIC netlist without modifications, e.g. FPGA optimization or special synthesis.

A library is required that contains translations of the ASIC library cells into FPGA primitives and gives information about the cell properties like complexity and IO connections to the partitioner.

4.3. Partitioning

The partitioning software is the key component of the software part in a prototyping system. It determines the number of signals required to interconnect all parts of the design. Due to the limited number of interconnections physically available within the prototyping hardware, this task influences the performance and the number of devices or boards required to implement the prototype.

Constraints for the partitioner are determined by limitations of the target architecture and devices in terms of IO count and number of available interconnections as well as equivalent gates.

After completion of the partitioning, the partitions must be allocated to the physically available FPGAs. In order to use a simple placement algorithm, it is important that the routing architecture is as symmetrical as possible. If there are any inhomogenities in the routing architecture the placement depends on the physical device chosen.

After placement, a global routing step is necessary that routes the interconnections between the different FPGAs. The connections to the outside world will also be routed in this step. The routing will then be implemented within the routing devices.

4.4. Compilation

After all logic has been placed and the global interconnections have been generated, the FPGAs and the routing devices can be compiled. For this purpose the device vendor tools are used which are interfaced by the prototyping system software.

The compilation takes some time depending on the utilization of the devices. The result of this process is a set of completely configured devices. Now the download bitstreams can be generated.

4.5. Download

Next the bitstream files have to be downloaded to the emulation hardware. In the case of the RP 2000 prototyping system a file transfer from the workstation to a PC is necessary because the host interface is PC based. After downloading the bitstreams into the emulator the design is ready to process data.

4.6. Prototype operation

In general there are two different options of data application to the emulated design.

The first option is to connect the prototype directly to the target system, i.e. the system that the circuit has been designed for. This is referred to as In Circuit Emulation (ICE).

The other possibility is to apply vectors from the controlling host and then capture the results produced by the emulation prototype.

The weakness of the second mode is that the data rate will be limited by the performance of the host interface. Usually this speed will be much lower than the maximum possible operating frequency of the emulator and therefore becomes a bottleneck.

In the case of the echo generator example both modes have been investigated.

Firstly the prototype has been connected to a system that has been setup to test audio applications as shown in Fig. 6.

![Fig. 6. Test system for audio applications](image)

This system consists of a digital data buffer. The DAT recorders can be controlled by the buffer system via RS422 interfaces.

For the echo generator the maximum required operating frequency was 48kHz. Because the emulation prototype runs at this frequency, the buffer is not necessary. The DAT has been interfaced by the prototype with some additional hardware for decoding the serial DAT format into a parallel format.
Secondly, a Sun sound file has been translated into the Zycad testvector format and then these vectors have been applied through the host PC. For a comparison of the runtimes refer to the next section.

5. Results

The goal of the presented example was to proof the feasibility of using existing products for a real high level design process.

For the demonstrated application, the proposed flow functioned satisfactory, even when some manual modifications in the different representations of the design were required. In the next section a speed comparison between the different verification methods for the testcase are presented.

5.1. Partitioning results

Table 1. lists the results of the partitioning process in terms of number of IOs and equivalent gates per partition for the main blocks of the design. The utilization is calculated based on 10000 raw gates for a XC 4010 device and the speed is estimated using Xilinx Xdelay tools.

<table>
<thead>
<tr>
<th></th>
<th>Reg</th>
<th>Weights</th>
<th>Adder</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td># IOs</td>
<td>17</td>
<td>45</td>
<td>73</td>
<td>55</td>
</tr>
<tr>
<td># Gates</td>
<td>5616</td>
<td>1272</td>
<td>205</td>
<td>1318</td>
</tr>
<tr>
<td>Util.</td>
<td>56 %</td>
<td>13 %</td>
<td>2 %</td>
<td>13 %</td>
</tr>
<tr>
<td>Speed</td>
<td>69.9MHz</td>
<td>3MHz</td>
<td>3MHz</td>
<td>20MHz</td>
</tr>
</tbody>
</table>

Table 1. Partitioning results

5.2. Setup and runtimes

The table below gives an overview about the setup times and runtimes of the different tools. The row for the setup time gives just a qualitative ranking of the four verification approaches.

<table>
<thead>
<tr>
<th></th>
<th>Cossap Sim.</th>
<th>behav. VHDL Sim.</th>
<th>Host vector apply</th>
<th>ICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup time</td>
<td>short</td>
<td>medium</td>
<td>medium-long</td>
<td>long</td>
</tr>
<tr>
<td>Runtime</td>
<td>6,3 min</td>
<td>~75 min</td>
<td>~823 min</td>
<td>31 sec</td>
</tr>
</tbody>
</table>

Table 2. Performance comparison

All software tests have been executed on a Sun Sparc 10 workstation. The data used for this comparison was a Sun recorded sound file with a length of 31 seconds. At a sampling rate of 48kHz there are approximately 1.5 million vectors that must be processed.

The first column shows the values for the Cossap DSP simulation tool. Internally, this tool works either with a C or FORTRAN implementation of the models. Therefore it has a short setup time and a long runtime. As VHDL simulator Vantage VSS has been used that is a compiled code simulator. The setup time for this simulation contains the creation of the testbench in VHDL and the format conversion of the test vectors into an ASCII format. The runtime is longer than with Cossap. The setup time for the emulation prototype in relation to the others is very long because a synthesis step is necessary, the resulting netlist must be partitioned and the FPGAs have to be placed and routed.

For the simulation mode, in which the vectors are driven by the host, the setup is then completed.

The long runtime for this mode in the table is caused by the limited speed of the host interface. For the example 82 bit wide input vectors were necessary. The speed of vector application was 30 vectors per second.

5.3. Possible Improvements

The presented prototyping design flow worked satisfactory but there are a couple of possible improvements for more efficient prototyping:

1. Avoiding the synthesis to gate level:

   The synthesis into a gate level netlist is step in the design flow where information that has been developed in the DSP design tool will be lost. The gatelevel netlist is read into the prototyping software and flattened out.

   If the synthesis tools would be able to map into more complex structures like adders or multipliers, it would be easier for the FPGA level tools to map the design into the chosen FPGA architecture.

   A proposal for a technology independent standard has been published by the EIA. This standard is called LPM (library of parameterized modules [11]) and describes a set of 25 different modules that have been identified to represent the most important building blocks for a wide range of digital designs.

2. Expansion of the bittrue libraries:

   The bittrue libraries of the DSP tool contain very simple blocks. It would be desirable to have more powerful functions that can be written out as VHDL code.
3. Resource sharing or scheduling techniques:
   As described before, the DSP design tool does not support any automatic scheduling or resource sharing. This makes the resulting prototype inefficient in terms of required FPGA resources for emulation. Therefore it would be helpful to get support in using techniques like described in [12] within the DSP environment.

4. Improved partitioning:
   The partitioning of the circuit is the step that determines the performance of the emulation prototype. If the partitioning is inefficient, e.g. critical paths are splitted among multiple devices, the FPGA tools cannot achieve the best possible performance for the circuit.
   If the partitioning is done after a VHDL level simulation like described here, simulation data could be used to find the critical paths and the most active signals of the design. This information could then be passed to the partitioner.

5.4. Conclusions
   There are three main conclusions of this work:
   1. Already available tools today can be used for rapid prototyping purposes.
   2. There are details to be optimized in order to achieve best results at a minimum of overhead for rapid prototyping.
   3. Rapid prototyping techniques can efficiently be used to increase verification depth as well as to shorten verification times of digital systems.

References