

Generated DSP cores for implementation of an OFDM Communication system

Hendrik Seidel, Emil Matus, Gordon Cichon, Pablo Robelly, Marcus Bronzel,
and Gerhard Fettweis

Mobile Communications Chair, TU-Dresden
D-01062 Dresden, Germany
seidel@ifn.et.tu-dresden.de

Abstract. Application tailored signal processors fill the gap between ASICs and general purpose DSPs. Single Instruction Multiple Data (SIMD) Signal Processors offer high computational power with low control overhead. This paper describes the development of a multi-processor OFDM-System x using automatically generated SIMD-DSP Cores. The focus of this case of study was the test of our integrated design flow which is based on our core generation tool. We show how with our design methodology we reduce the design cycle in comparison with other HW/SW Co-design tools and traditional design flows.

1 Introduction

Signal processing applications for high end communication devices require two different types of application specific signal processors: The first type is a highly optimized DSP core with a complex instruction-set to handle the demanding application specific computational requirements. The second type is a more generic small processor core which can be used with different application specific cores or as a stand-alone-core for low data rate applications. Powerful tools are needed to minimize the time needed for designing and validating these DSPs. Therefore, the required software toolchain should be generated together with a processor core which enables efficient reuse of previously created hardware units. This can be achieved using a dedicated hardware library together with a capable architecture template which facilitates short and frequent design cycles.

The STA (Synchronous Transfer Architecture) [2] is such a template. A compiler-friendly machine description describes DSP cores. Then, the machine description is used by GenCore [1] to create hardware and simulation models of application tailored DSP cores. Compared to the more general processor design tools like CoWare/LisaTek [7], PEAS-III [5] and EXPRESSION-ADL [4] which are more targeting RISC and CISC architectures, GenCore is focusing on SIMD vector DSP architectures. In our case of study we have developed an OFDM broadcasting system using an STA-I/O Processor and an STA SIMD DSP on receiver and transmitter side.

2 Case Study: OFDM

OFDM is used in numerous digital communication systems: DVB, ISDB and DAB for digital broadcasting, IEEE802.15.3a and IEEE802.11a for wireless networking and ADSL modems for wireline communication. These systems are transmitting IP, video and audio streams with data rates up to multiple Mbit/s. High performance ASICs [3] and multi-DSP solutions are used for embedding these standards into set-top-boxes, PCMCIA-cards or mobile terminals.

In our first design we have developed an initial platform to compute the basic algorithms of these standards. The primary goal was to demonstrate the feasibility of our generated platform concept in order to achieve short design cycles. In a first case study we have developed a simple OFDM based communication system with transmitter and receiver. Both comprise two different STA based DSP cores which have been integrated on a single FPGA: The generic STA-I/O Processor and STA-OFDM SIMD DSP. For a rapid prototyping of the system, the design is implemented on two Stratix25 FPGA development boards (shown in figure 1) with ADCs and DACs for baseband signal processing. The input and output of the OFDM system consists of an uncoded data stream which is transferred via a USB 2.0 interface to a host PC for post processing.

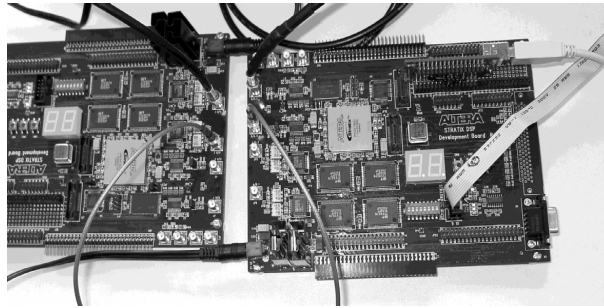


Fig. 1. Development Board Setup

2.1 Functional Blocks

For the OFDM System, we implemented functional blocks similar to those being used in DVB-T/H broadcasting systems. Each functional block at the transmitter has a corresponding block at the receiver as shown in figure 2. At the transmitter the Input Data from a Host-PC is mapped to QPSK-symbols. After insertion of BPSK pilot carriers (Barker Sequence) the IFFT (with decimation in frequency) is calculated. The resulting time domain signal is finally converted into an analogue signal using a dual DAC with 500kHz sampling frequency. At the receiver the incoming symbols are oversampled at 2MHz with a dual ADC.

Depending on the cross-correlation results after the FFT, the phase of the incoming stream is shifted by the synchronisation unit. Every fourth symbol is stored into the memory. After calculation of the FFT, the BPSK pilot signals are cross-correlated with the barker sequence and the complex result is stored for later use in the synchronisation unit. Following the detection process, the received data is sent to a Host-PC. [6] The OFDM parameters are shown in table 3. For the first implementation the sample rate was set to 500kHz, but 3-4 times higher sample rates are possible as shown in paragraph 4.

3 SoC Development

After analyzing the system requirements and the underlying algorithms a first draft of the SoC concept was designed. In order to guarantee a short time frame for design and testing, we decided to reduce custom hardware design to a minimum and generate as much as possible automatically from our core generation tools. Instead of having a communication bus between the processors and the peripherals, we devised a small automatically generated STA-processor as DMA controller, IO processor and task scheduler. This OFDM-IO processor manages all data transfers. The SIMD OFDM-DSP signal processor was added to the OFDM-IO as a Co- Processor. The OFDM-DSP computes all signal processing algorithms described in paragraph 2.1. It provides an interface to its scalar memory. A program halt is indicated by a control flag. The sequencer of the OFDM-DSP is controlled by the OFDM-IO processor. With machine descriptions of both processors, two instances of the STA- architectural template were generated using GenCore. Many VHDL¹-components available from previous designs could be reused for this OFDM SoC. Only a few new components had to be added to the design. To ensure that the design fits on the 25k cells of the FPGA, a rough cell-count estimation was performed. The whole design was synthesized and routed without doing any previous functional tests of the system to get an estimate of the expected cell area for the final design. Software and hardware development were started with the first release of our SoC. Figure 4 provides an overview of our SoC design flow.

¹ Very High Speed Integrated Circuit Hardware Description Language

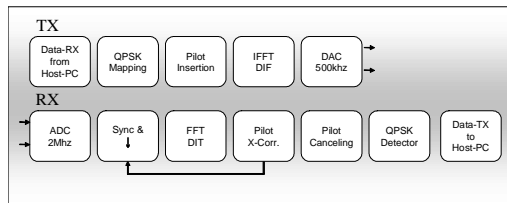


Fig. 2. OFDM Transmitter and Receiver (Block Diagram)

Total number of carriers	64
Data carriers (QPSK)	48
Pilot carriers (BPSK)	7
Barker-Code	
Bits per OFDM symbol	96
Sample rate	> 500kHz

Fig. 3. OFDM parameters

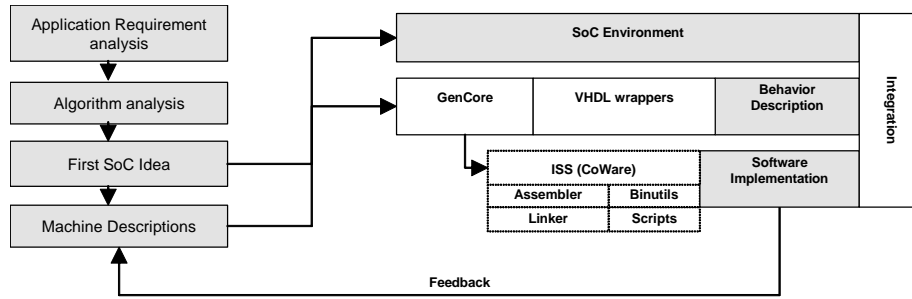


Fig. 4. Integrated Design Flow with GenCore

3.1 Processor Design

STA with GenCore: The core generation tool GenCore creates instances of the synchronous transfer architecture template. In STA, a functional unit is defined as module with output registers and input multiplexers. The input multiplexers of each functional unit are connected to one or more output registers of other functional units. Depending on the application, power, area and performance constraints, this multiplexer network is more or less complex. Functional Units perform computation. Register files and memories store states.

To describe a STA-based processor, a special machine description format is used. The format is called RNA and is based on annotated graphs [8]. The machine description is subdivided into 6 major sections:

- Types
- Toplevel
- Register Files
- Memories
- Functional Units
- Operations

A LISA[9] model and a VHDL framework² are generated based on the STA-architecture template and machine description. The VHDL framework consists of memories³, register files, an instruction decoder, a multiplexer network and functional unit VHDL-wrappers. The VHDL-wrappers have to be filled with a behaviour description of the functional unit. With every new operation the multiplexer network and decoder are adjusted automatically. Implementation errors are minimized by having just a single VHDL-file to be changed by hand if new instructions are added.

² The VHDL framework is generated by GenCore, not by CoWare Tools

³ only ALTERA memories are supported

Design Space Exploration: STA-processor design is done on three abstraction levels:

1. Instruction Set Simulation with the CoWare LISA ISS Debugger for software development and cycle count optimization.
2. VHDL-Model Simulation using Synopsys/Scirocco for hardware debugging. (For silicon implementations in this layer performance, area and power estimations are done)
3. FPGA prototyping to test the processor with other hardware devices and to run tests like bit error rate calculations.

The ISS is used as reference for the other abstraction levels. Therefore, the first step in the hardware debugging process is the comparison of VHDL-model and FPGA-prototype with the ISS.

	ISS	VHDL-Simulation	FPGA-prototype
Simulation time	Medium	Very High	Very Low
Debugging Information	High	High	Very Low
Coverage	Software	Software/Hardware	Software/Hardware
Cost	Low	Low	Low - Medium

Table 1. Comparison of the test and development platforms of STA-processors

In table 1 the different simulation and emulation methods are listed. Adequate debugging methods could be chosen from this set.

3.2 OFDM-DSP

The OFDM-DSP is a fixed point SIMD parallel signal processor (A block diagram is shown in Figure 5) with a high degree of instruction level parallelism. The vector part of the DSP consists of 4 parallel data paths. Each data path consists of an ALU, a multiply and accumulate (MAC) unit, a barrel-shifter (BS) unit, a register (REG) file, a conditional unit (COND) and a conditional data transfer (IF)-unit. With IF units and COND units, the DSP is able to parallelize conditional data transfers. The Scalar data path of the processor consists of two ALUs, a MUL, a REG, a BS, a sequencer (SEQ) and a COND unit. The data path is principally involved with address calculations and program control. In the vector part of the processor, the COND unit is connected to the IF-Unit for data flow control. In the scalar part, the COND unit is connected to the sequencer for conditional branches. The processor has three single port memories: a program Memory, a scalar memory and a vector memory. These Altera Memories were generated automatically by GenCore. The bitwidth of all interconnections is 16 bits. However, there is a single exception: a 32 bit connection between the Accumulator of the MAC Unit and the barrel shifter unit, helps to increase computation accuracy.

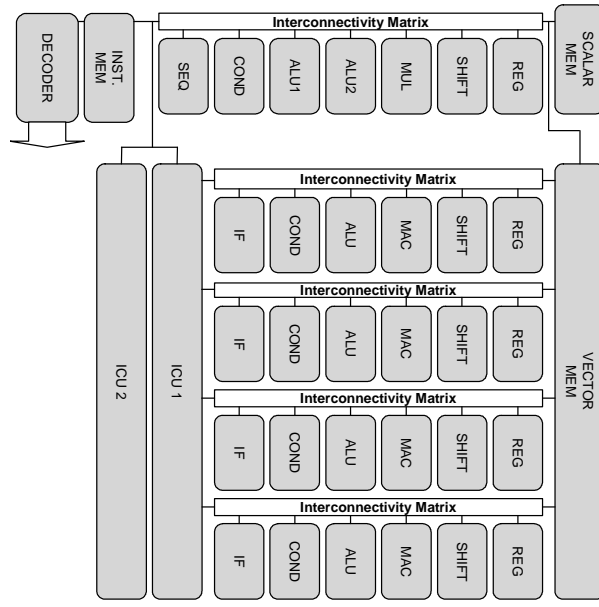


Fig. 5. OFDM-DSP Block Set

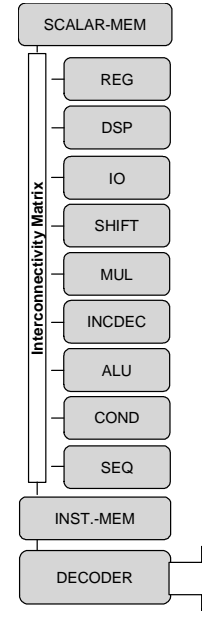


Fig. 6. OFDM-IO Block Set

3.3 OFDM-IO

The OFDM-IO processor, as shown in figure 6, is similar to the scalar data path of the OFDM-DSP. Instead of the second ALU, an increase and decrease unit (INCDEC) was implemented to enable simple address calculation. The Instruction Level parallelism of STA-processors enables efficient data transfers between Functional Units and Memories. The OFDM-DSP and the I/O ports are connected to the OFDM-IO within functional unit -wrappers. Hence, both units have input multiplexers and output registers like standard functional units e.g. an ALU or an MAC. Specialized instructions to write or read the ADC/DAC and to access the USB 2.0 Interface exist. Data exchange with the OFDM-DSP is done via direct memory access.

3.4 Environment

The OFDM-IO processor is embedded into the operational environment (figure 7). The environment of the processor consist of the:

- input and output baseband signal channels,
- input and output data (RX/TX) channels,
- debug interface,
- debug input and output signal channels,
- clock generator.

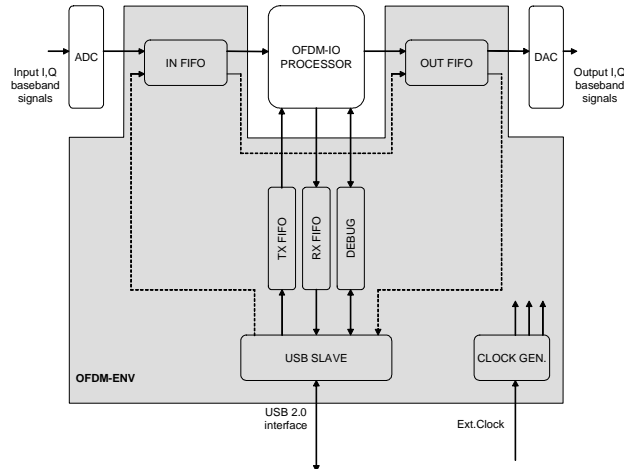


Fig. 7. SoC Environment

PC, as master, controls the OFDM transceiver over the general purpose USB 2.0 interface. USB slave controller is located in the transceiver. The data RX/TX channels and debugging port are implemented in the USB slave controller. Optionally also input/output baseband signal channels could be used to emulate ADC and DAC interfaces. Debug interface enables address, data in/out and control signal generation. All memory resources of both IO and DSP processors are accessible at any time over debug interface. Thus, the processor memory down/up load from/to PC is possible. In the receiver mode the input I,Q baseband signals are digitized by ADCs and stored to the input FIFO buffers. The OFDM-IO processor reads the signal samples to be processed from input FIFO buffer and serves them to the DSP processor. The demodulated data is send to RX data channel and received by host over USB interface. In the transmitter mode, the data to be transmitted is send by host to TX data channel over USB interface. IO processor reads the data from TX FIFO buffer and passes them to OFDM-DSP. Output baseband signal generated by OFDM- IO processor is stored to ouput FIFO buffers. Output FIFOs are directly connected to DA converters to produce analogue baseband signal.

4 Results

The presented design was implemented within 5 weeks by two engineers. The processor design and its implementation were finished after three weeks. The main hardware implementation effort was spent on the environmental devices like the USB 2.0 Interface and converters. Software development was started

in the second week of the development process after the LISA description was generated from the first version of our STA- machine description.

Hardware Description: In Table 8 the complexity of STA-MD code is compared to the generated LISA and VHDL Code. Also in consideration of the fact that a handwritten LISA or VHDL model is smaller than our generated models, a STA-machine description is less complex than LISA and VHDL description.

To represent the implementation effort for each of the processors we have calculated the lines of code of the VHDL-models. The comparison is shown in figure 9. We distinguish between generated code, handwritten code and code taken from a hardware library. We assume that we have to build the hardware library from scratch. If we look at both processors, the code taken from a hardware library increases. A next implementation based on this design will have even more code taken from a hardware library. This high reusability is based on the fact that the STA-architecture is extremely structured.

Hardware Implementation: Synthesis and Place and Route were done with the Altera Quartus II Software without any manual optimization or detailed timing constraints. The OFDM-IO consumes 5179 cells, the OFDM-DSP 16686 cells and the environmental devices 575 cells. Overall 85% of the 25k FPGA cells were used. A detailed partitioning of the cells for the OFDM-DSP is shown in Figure 10. Most of the Cells were spend on functional units and the multiplexer network. The control overhead of the DSP is only 4%. The current size of the multiplexer network compared to the functional units is alarming. For a redesign we have to decrease the number of available connections. For STA-based designs a simple profiling tool is able to detect not frequent or never used multiplexer states. A complexity decimation of 50%-75% could be reached with little effort.

Software Implementation: In figure 11 and 12 the Clock Cycles for each algorithm are presented. Mapping, Detecting, Pilot Insertion and Removal take only a little percentage of the overall computational effort. The main parts are IFFT/FFT and Memory Transfers. The memory transfers can be integrated into

Lines of code	IO-P	DSP
STA-MD	619	1389
LISA	11007	27892
VHDL	2592	4784

Fig. 8. Generated ADL/HDL-Code from STA-machine description

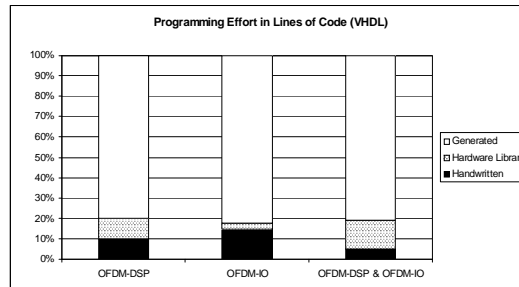


Fig. 9. Programming Effort

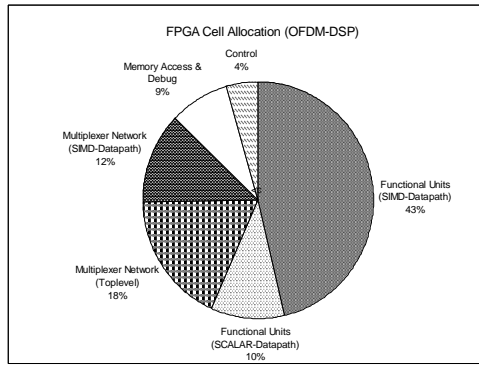


Fig. 10. FPGA Cell Allocation

the FFT and IFFT calculations, hence FFT and IFFT remain as major part of the whole computation. The IFFT (DIF) was implemented after the FFT (DIT). Thus, the developer was more familiar with the architecture as he programmed the FFT a second time. As the theoretical limit for the FFT with a 4 Slice SIMD parallel DSP with one MAC-Unit in each slice and a single port data memory is about 192 clock cycles, the current implementation has high optimization potential. About 500 clock cycles for FFT and 1000 clock cycles to compute a complete OFDM-frame is feasible. Therefore, the system is able to transmit or receive with a core clock rate of 40MHz at 2.5Mhz sampling frequency. This results in a maximal transmission rate of 3.84Mbit/s.

To use this system for real applications, the synchronization has to be more sophisticated and forward error correction and equalization techniques have to be implemented. A RF-frontend has to be added, as well.

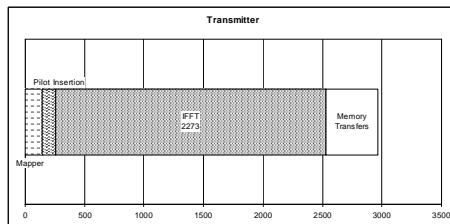


Fig. 11. Clock Cycles Transmitter

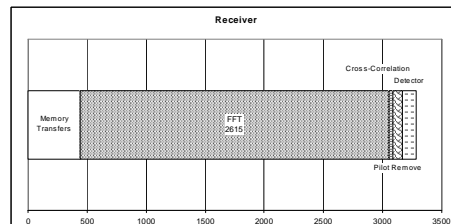


Fig. 12. Clock Cycles Receiver

5 Conclusion

Our integrated design flow which is based on our core generation tool is suitable for fast VLSI prototyping. Common implementation failures as well as effort could be reduced by generating most of the source code automatically. Special features for automatic SIMD-processor generation allow for enlarging the design with little effort. The OFDM Transmitter and Receiver have been developed within five weeks by two engineers. Test and Verifikation is simplified by the structured STA-approach and the high hardware reusability.

With further software optimization, uncoded OFDM Transmission with a data rate up to 3.84Mbit/s is possible. A weak spot of the current architecture implementation is the large multiplexer network. This problem has to be eliminated by removing not used interconnections in future.

In consideration of the fact that we spent most of the development time on embedding the environmental devices, design tools have to be developed to face this problem.

6 Acknowledgements

The authors would like to thank Martin Goblirsch, Markus Ullmann und Markus Winter for valuable contributions to this project and Winnie-Kathrin Ahlendorf for realizing XP.

References

1. Gordon Cichon, Pablo Robelly, and Hendrik Seidel. *STA Core Generator Manual*. TU-Dresden, 2003. www.radionetworkprocessor.com.
2. Gordon Cichon, Pablo Robelly, Hendrik Seidel, Marcus Bronzel, and Gerhard Fettweis. Synchronous transfer architecture (sta). In *Proc. of Fourth International Workshop on Systems, Architectures, Modeling and Simulation (SAMOS'04)*, Samos, July 2004.
3. Babak Daneshrad, Jr. Leonard J. Cimini, and Manny Carloni. Clusterd-ofdm transmitter implementation. In *Proc. IEEE PIMRC'96*, pages 1064–1068, Taipei, Taiwan, Oct 1996.
4. A. Halambi and P. Grun. Expression: A language for architecture exploration through compiler/simulator retargetability, 1999.
5. Akira Kitajima, Makiko Itoh, Jun Sato, Akichika Shiomi, Yoshinori Takeuchi, and Masaharu Imai. Effectiveness of the asip design system peas-iii in design of pipelined processors. In *Proceedings of the 2001 conference on Asia South Pacific design automation*, pages 649–654. ACM Press, 2001.
6. John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing*. Prentice Hall, third edition, 1996.
7. CoWare LISATek prodcut family. Automated embedded processor design and software development tool generation.
8. *RNA*. <http://www.rna.cichon.com>.
9. Vojin Zivojnovic, Stefan Pees, and Heinrich Meyr. Lisa - machine description language and generic machine model for hw/sw co-design, 1996.