Application Specific Instruction Processor Based Implementation of a GNSS Receiver on an FPGA

G. Kappen, T. G. Noll
RWTH Aachen University, Chair of Electrical Engineering and Computer Systems, Schinkelstr. 2, 52062 Aachen, Germany
{kappen , tgn}@eecs.rwth-aachen.de

Abstract

In this paper the concept of a reconfigurable hardware macro to be used as a generic building block in low-power, low-cost SoC for multioperable GNSS positioning is described, featuring sufficient computational power and flexibility. The central processing unit of the reconfigurable hardware macro is an ASIP accelerated by additional eFPGA and weakly configurable ASIC based co-processors. The different hardware building blocks (i.e. ASIP, eFPGA, ASIC) of the target architecture are motivated with state of the art GNSS receiver algorithms. To explore the design space of the target architecture and to develop appropriate partitioning cost functions a GNSS receiver testbed was realised on an FPGA board. The testbed utilises a programmable ASIP, designed and generated with the processor description language LISA, as a central processing unit. As a first accelerating co-processor the correlator was realised. Exemplary optimisations of the ASIP / co-processor architecture as well as the achieved improvements are described.

1. Introduction

Today there is an increasing variety of applications for global navigation satellite system (GNSS) receivers ranging from simple navigation equipment or location based service in mobile phones to global time reference applications.

In near future with the existing American Navstar GPS, the Russian GLONASS and the upcoming European GALILEO system, three GNSSs will be available. In addition, the user has the opportunity to use so-called augmentation systems like the European EGNOS to receive additional positioning, integrity and correction information. The interoperable use of these systems will improve the accuracy as well as the availability and supports GNSS real-time kinematic (RTK) positioning [1].

RTK and multioperable positioning, as well as applicability to a wide range of applications will require high computational performance. A high degree of flexibility is demanded because of continuous evolution in the GNSS receiver algorithms and expected signal specification changes for GALILEO and the modernized Navstar GPS. Furthermore, most of the battery-powered applications are power dissipation and/or energy critical, leading to the well-known "power vs. flexibility conflict".

In this paper the concept of a reconfigurable hardware macro to be used as a generic building block in low-power, low-cost System-on-Chip (SoC) is presented. The proposed target architecture (Figure 1) allows for multioperable GNSS positioning, simple integration in SoC and features sufficient computational power and flexibility. The target hardware architecture comprises an analog frontend with an A/D-converter and a digital signal processing part built up from a programmable application specific instruction processor (ASIP) being accelerated by application domain specific embedded FPGA (eFPGA) and weakly configurable ASIC based co-processors.

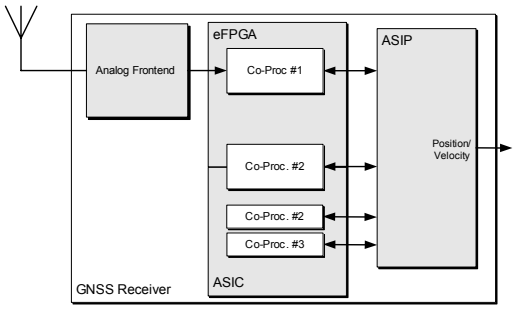

Figure 1: Outline of the target architecture

The paper is organised as follows:

In the second section the architecture of standard GNSS receivers and the realisation of the main signal processing blocks will be sketched. In section 3, based on flexibility and performance constraints, a first partitioning is derived and the accelerating blocks of the target architecture are motivated. In the fourth section the testbed implementation of an ASIP-based multioperable GNSS

3-9810801-0-6/DATE06 © 2006 EDAA

receiver on an FPGA board is presented and first optimisations of the ASIP instruction set are outlined.

2. Standard GNSS receiver

This section gives a brief overview of the architecture of a standard GNSS receiver and introduces the main signal processing blocks. For a detailed description of GNSS receiver applications, operation and algorithms refer to [2].

Today's standard GNSS receivers comprise an analog and a digital part. The analog part consists of the analog frontend transferring the incoming RF signal to an intermediate frequency (IF) and converting it from analog to digital. For NAVSTAR GPS and GALILEO, the satellite signals use the same carrier frequency and are separated in the code space domain (CDMA) by so-called pseudo random noise (PRN) codes. GLONASS separates the satellites in the frequency domain (FDMA) which requires a more complex RF frontend architecture. For a multioperable receiver the frontend should consider the RF signals from all available GNSS (e.g. GPS-L1, GLONASS, GALILEO L1/E5a, etc.).

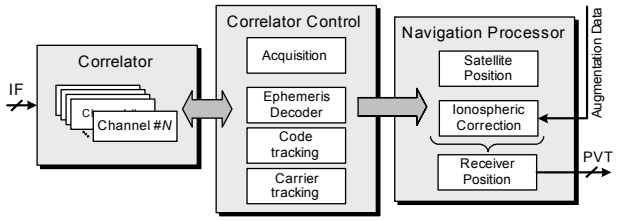

Figure 2: Simplified GNSS receiver block diagram

The digital part (Figure 2) consists of *N* parallel correlator channels, the correlator control block and the position velocity time (PVT) calculation. The three blocks are described in the following.

Figure 3 shows a Navstar GPS correlator channel in detail. In the first stage the IF signal is multiplied with the complex local estimate of the intermediate carrier frequency.

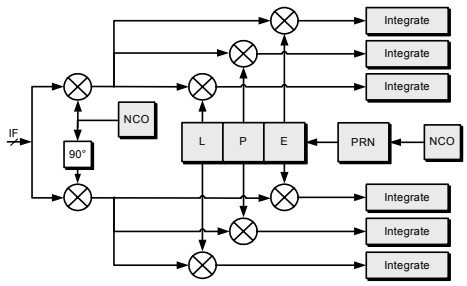

Figure 3: GPS Correlator Channel Block diagram

The resulting complex baseband signal is multiplied with three different, half-chip separated, versions of the PRN code (i.e. late (L), prompt (P), early (E)) and integrated and dumped in six correlation registers. These six registers form the input values for the carrier and code tracking. GLONASS and GALILEO correlators differ from the Navstar GPS correlator (Figure 3) mainly in the used PRN code. GALILEO additionally uses a modified modulation scheme called Binary Offset Carrier (BOC) which modulates the PRN code with a sub-carrier [3].

In the correlator control block, code tracking aligns the code phase of the incoming and locally generated PRN code by a delay lock loop (DLL). The DLL input values are the early and late correlation registers. Carrier tracking utilises the prompt correlation values to adjust the carrier frequency with a combination of a Phase-Locked Loop (PLL) and a Frequency-Locked Loop (FLL). The ephemeris decoder extracts the ephemeris data (i.e. satellite orbit data, correction parameters, etc.) out of the satellite signal.

The navigation processor calculates the satellite and the receiver position for a specific time, based on the decoded ephemeris and the measurement data of the correlator channels. Furthermore the PVT refines the positioning results by correction of the estimated ionospheric delay based on ephemeris and/or additional augmentation data (e.g. EGNOS).

In today's standard Navstar GPS receivers the correlator channels are realised in dedicated hardware and thus the channel number and configuration is fixed. The correlator control and the navigation processor are mostly implemented on an embedded RISC processor (e.g. [4]).

In the following section, starting from a description of the flexibility and performance requirements of the signal processing blocks (Figure 2), the hardware architecture (Figure 1) of the proposed GNSS receiver macro is motivated. Hereafter a first mapping of the signal processing blocks to the target architecture is done.

3. Target architecture

3.1 Motivation

To improve area and power efficiency the signal processing blocks (Figure 2) should be mapped to the appropriate hardware block of the proposed reconfigurable hardware macro. Therefore the flexibility and performance requirements of current and next generation GNSS signal processing must be analysed first:

In order to improve the accuracy and the sensitivity of the receiver currently intensive research is undertaken to elaborate new concepts in the field e.g. of multipath propagation and cross correlation mitigation.

Multipath propagation effects arise from the superposition of the direct and delayed path signals and directly

affect code tracking and though the positioning accuracy due to deformation of the received signal.

Recent publications propose new correlator configurations to overcome the problem of multipath propagation. In [5] and [6] the above mentioned early / late correlator code tracking is expanded to a solution where linear combinations of early and late correlators with flexible delay, so-called Code Correlation Reference Waveforms (CCRW), are used. Other approaches use the correspondence between the received signal power and the multipath error [7] to especially account for short delay multipath propagation. Concerning the new modulation scheme of the GALILEO signal other approaches for enhanced correlators in [8] and [3] are proposed.

Furthermore, recent publications in the field of cross correlation mitigation for high sensitivity receivers [9] and enhanced tracking algorithms proposed in [10] require further changes in the correlator channel.

In conclusion, today it seems to be impossible to specify a dedicated correlator channel enabling for the implementation of future enhanced correlation concepts. Even more severe it is expected that the most appropriate correlation concept in future will strongly depend on the addressed application domain. Furthermore, expected signal specification changes for GALILEO and the modernised Navstar GPS, as well as different correlator channel structures for the available GNSS support this approach.

The acquisition algorithm allocates the observable satellites to the receiver channels. Depending on the available data (Almanac, Ephemeris, Assisted GPS) there is some kind of a priori knowledge which can be exploited. In addition, different search strategies (e.g. frequency domain, serial search) could be used, which affect the time for a first position information (i.e. time-to-first-fix).

The tracking adjusts the code and carrier frequency after the satellite signal has been detected and locked. The tracking loop usually consists of a discriminator and a loop filter. For best tracking results the parameters of the tracking loop algorithm (e.g. order / bandwidth of the loop filter, type of discriminator, etc.) should be adjustable in real-time to consider varying receiver conditions (e.g. temporal high receiver dynamics, weak satellite signals, etc.).

The navigation processor mainly performs trigonometric and matrix computations to estimate the receiver and satellite position and correct the ionospheric delay. To decrease the effect of noisy measurement data or to integrate inertial navigation sensors (INS) in different stages [11] of the receiver signal processing, the Kalman Filter is frequently used in the navigation processor block of GNSS receivers. Through the extensive use of matrix operations and a great number of state variables the Kalman Filter requires a high computational performance.

After the analysis of the requirements of the signal processing blocks in this section, the next section describes the proposed hardware architecture.

3.2 Hardware building blocks

The central processing unit of the proposed multioperable GNSS receiver platform will be realised as an ASIP optimised for processing of GNSS receiver algorithms with minimal power and area consumption [12]. In order to achieve the required computational performance at an acceptable overall power and area efficiency, the ASIP will be accelerated by additional tightly coupled building blocks (co-processors).

To motivate this target architecture, Figure 4 illustrates the design space in terms of power efficiency (in mW/MOPS) and area efficiency (in MOPS/mm²). The different entries in the diagram correspond to implementations of standard signal processing algorithms on different hardware platforms (e.g. General Purpose (GP) processor, DSP, FPGA, etc.). The entries are clustered according to the respective hardware platform. The flexibility decreases from left to right (programmable, reconfigurable, dedicated). A standard GPS correlator has been implemented on three different platforms (GP-processor, DSP and FPGA) to approve the diagram for the GNSS receiver case. The correlator implementations are depicted by red symbols in the diagram. As can be seen from Figure 4 ASIPs and eFPGAs establish an attractive compromise between standard programmable cores on the one hand side and dedicated hardware cores on the other side.

Dedicated hardware implementation offer superior power and area efficiency but suffer from the fact that they feature no flexibility at all.

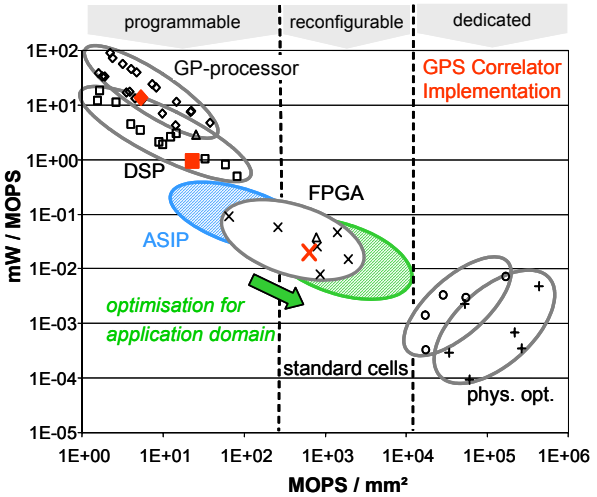

Figure 4: Power vs. Area Design space

As the area efficiency of eFPGA implementations is about one to two orders of magnitude worse than that of

dedicated hardware implementations the use of multiplexed dedicated hardware features the best solution as long as the number of different modes is limited and especially if the functionality of the different modes can be fixed during design time. Even if the number of modes gets large (and area efficiency decreases) this approach ensures lowest possible power dissipation by simply switching off non-used units.

Of course, non-iterative and irregular control oriented functionality calls for obvious reasons for a software implementation on programmable cores. Therefore, here a hybrid architecture is proposed (Figure 1):

Control oriented functionality and positioning calculations are to be mapped on a software programmable processor core. In order to achieve best possible power and area efficiency instead of applying a standard processor core an architecture being specifically to this application (ASIP) will be applied. Number crunching functionalities which can be specified as generic sub-functions are implemented on multiplexed dedicated hardware accelerator blocks. Units (especially parts of the correlator block) which neither can be specified finally nor are suitable for an implementation on the ASIP will be mapped on a reconfigurable eFPGA-based accelerator block. In order to significantly improve the power and area efficiency of the latter, its structure (especially its interconnect architecture) will be optimised for this application towards an application specific eFPGA (Figure 4).

One example for an application domain eFPGA optimisation is the reduction of area an especially power consuming medium length interconnect resources [13]. This modification is based on the observation that regular in contrast to irregular logic requires less medium length interconnect resources. To confirm this for GNSS applications, the Navstar GPS correlator (regular) and the correlator control (irregular) have been implemented on a commercial FPGA. For the two implementations the time delay between two nodes, which corresponds to the connection length, has been compared. Figure 5 shows the interconnect length histograms according to the qualitative assumption and the quantitative confirmation by this experiment. It can be seen that the correlator control logic needs a significantly higher fraction of medium interconnects than the regular correlator.

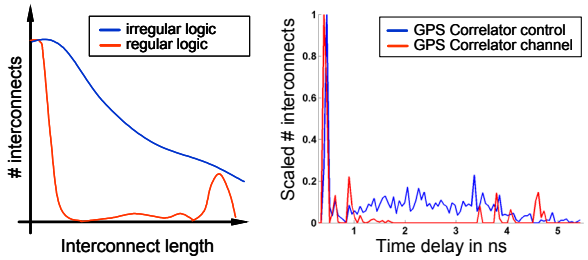

Figure 5: Connections lengths comparison

The optimisation of the ASIP is mainly done by customisation of the instruction set architecture (ISA) and the processor architecture for this specific application. The basis for the modifications of the ASIPs ISA will be the profiling of actual and upcoming GNSS receiver algorithms.

3.3 Partitioning approach

From the discussion above, in this section the mapping of the main signal processing blocks onto the hardware building blocks will be described.

Because a fixed correlator channel architecture is currently not conceivable, some blocks of the correlator especially the code generator and the CCRW generation will be implemented on a reconfigurable logic while other parts are realised on the weakly configurable ASIC.

In contrast, the acquisition is suited well for software implementation though this part should be implemented on the ASIP.

For the tracking it must be investigated if a (partial) mapping to the reconfigurable co-processors increases the overall receiver performance and energy efficiency without reducing the required flexibility of the tracking loop.

However, especially the loop filters of the tracking loop are suitable for the current eFPGA topology [13].

In the presented approach, the functions of the navigation processor will be implemented on the ASIP. Here, further research must prove the use of an application optimised instruction set for matrix calculations. The importance of this optimised instruction set is further increasing for the usage of the Kalman Filter for position smoothing and INS aided receiver operation. Further sophisticated algorithms could benefit from an instruction set, optimised for matrix computations (e.g. subspace projection methods [9] which reduce the cross correlation in high sensitivity receivers).

The different signal processing blocks of a GNSS receiver mentioned above, their need for flexibility and performance are considered to formulate a first coarse partitioning onto the different architecture blocks (Table 1).

Table 1: GNSS receiver partitioning

Signal Processing Block		Hardware Block
Correlator	→	eFPGA / ASIC
Acquisition	→	ASIP
Tracking	→	ASIP / eFPGA
Navigation Processor	→	ASIP

In the next section, the first implementation of a testbed on an FPGA board is discussed.

4. GNSS receiver testbed

For the purpose of target architecture evaluation and optimisation as well as exploration of approaches for multioperable positioning, a GNSS receiver testbed (Figure 6) was realised. Furthermore, the testbed will enable the formulation of cost functions for different partitioning scenarios.

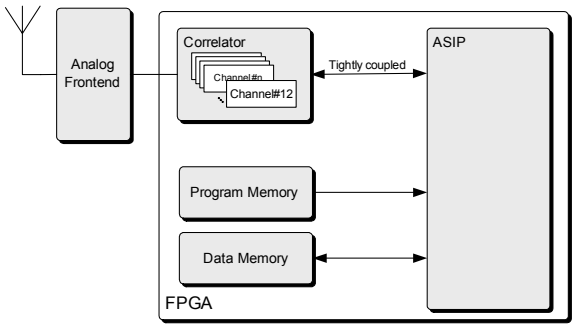

Figure 6: GNSS receiver testbed

The analog RF part of the testbed was realised by commercially available frontend and A/D components for all three GNSSs, while the digital part was implemented on a commercial FPGA.

The description of the ASIP and the instruction set was done using the processor description language LISA [14]. After generating the ASIP simulator model, the LISA Processor Simulator reduces the debugging effort because both the processor and the application code can be completely simulated before tested in hardware.

4.1 Prototype hardware

To have a first starting point for the ASIP a standard RISC processor LISA description has been modified to be synthesisable for a commercially available FPGA. The ASIP VHDL code was automatically generated by the LISA Processor Generator. The complete ASIP with data and program memory was synthesised and compiled by FPGA design software. Characteristic features of this ASIP implementation on a Stratix II FPGA are summarised in Table 2. To couple the ASIP to the accelerating co-processors tightly additionally data and control I/Os have been added to the ASIP architecture.

The remaining part of the FPGA is used for the reconfigurable implementation of the co-processors blocks.

Table 2: Characteristic ASIP values

#LEs	f_{max}	Data Memory	Prog. Memory
1765	112 MHz	4 kByte	4 kByte

The complete software development for the ASIP was done in assembler. After debugging the software, the generated executable files were converted to program memory content which could be loaded onto FPGA memory.

As a first example for a co-processor, a 12-channel correlator was implemented in VHDL. Each channel can be configured for Navstar GPS, GALILEO and GLONASS operation. Because of the tightly coupling between the ASIP and its co-processors, the ASIP can directly access the correlator measurement data and control the local carrier- and code-frequency of every correlator channel. Therefore, no communication protocol is required which significantly reduces the complexity of the ASIP software.

Table 3: Characteristic prototype values

#LEs	f_{max}	Data Memory	Program Memory
5557	84 MHz	4 kByte	4 kByte

In Table 3, the features of the testbed (digital part only) for the exemplary case with 12 correlator channels on a Stratix II FPGA are summarised.

4.2 Correlator control software

As a starting point for the ASIP software, the correlator control (i.e. acquisition and tracking) will be presented to show the co-processor / ASIP coupling and first ASIP optimisations. The correlator control software initially assigns the observable satellites to the correlator channels and starts a search in the code phase and frequency space for satellite signals. If satellite signals are present, the correlator unit synchronises the local carrier- and code-frequency to track these satellites. If the correlator control tracks a minimum of four satellites, the software reads the measurement result registers and decodes the ephemeris data to calculate the satellite and receiver position.

Benchmark results (Figure 7) of the complete GNSS algorithm reveal that especially the tracking takes a reasonable amount of the overall computational time, depending on the number of tracked satellites.

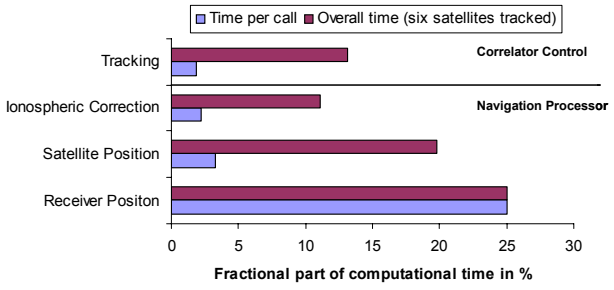

Figure 7: GNSS algorithm Benchmark results

4.3 Optimisations

First optimisations are presented to show how special instructions can significantly reduce the runtime of an application. In addition to this, the described modifica-

tions of the ASIP ISA show two different methods of supplementing new instructions in the coding tree and how this affects the ASIP features.

To show these modifications, the tracking is examined closer. On the standard RISC processor, the tracking runs 167 cycles.

The arithmetical right shift (`asr`) is a very simple operation which is added to the existing coding tree [14]. Because just a new leaf in the execute stage of the pipeline is added and the realisation in VHDL is simple, the total amount additional LEs is small. Nevertheless, the new instruction saves some execution cycles in the tracking routine code

The second exemplary modification concerns the calculation of two complex vector lengths (`vlen`) in one cycle. This instruction has been implemented as a new coding tree in the ASIP. Because the changes affect three pipeline stages and the realisation in VHDL is more complex, the number of LEs is significantly increased.

The results of the presented modifications are summarised in Table 4. With the area (A) of the ASIP and the execution time (T) of the ASIP software it can be seen that the exemplarily chosen efficiency $A \cdot T$ complexity decreases for the added instructions.

Table 4. Optimisation results.

	#LE	Cycles	f_{max}	AT
RISC	1765	167	112 MHz	**100 %**
+asr	1798	155	110 MHz	**96.2 %**
+vlen	2551	102	113 MHz	**87.5 %**

Further modifications will concern the instruction set of the ASIP as well as the coupling between the ASIP and the associated co-processors. The aim of the optimisations will be to achieve the performance specifications of the GNSS algorithm and then improve the area and power efficiency.

5. Conclusion

The concept of a reconfigurable GNSS receiver architecture based on an ASIP was presented. The proposed hardware uses an ASIP optimised for GNSS receiver algorithms which is accelerated by application domain optimised eFPGAs and dedicated weakly configurable ASIC blocks. The different signal processing blocks of actual and forthcoming GNSS receivers have been examined in terms of the needed flexibility and performance and a first partitioning has been presented.

A GNSS receiver testbed comprised of an ASIP and VHDL macros was realised on a commercial FPGA to explore the design space of the proposed reconfigurable target architecture. With this prototype actually the parallel reception and decoding of signals from up to 12 satellites is possible. First optimisations of the ASIP core and their effect on the computational time and the ASIP area were described.

Further research will focus on the optimisation of the ASIP ISA, the coupling between the ASIP and the accelerating co-processors and the optimisation of the eFPGA for GNSS.

References

[1] Eisfeller, B., Tiberius C., Pany T., Heinrichs G., Real-Time Kinematic in the Light of GPS Modernization and Galileo, GPS World, October, 2002

[2] Kaplan, Elliot D., *Understanding GPS Principles and Applications*, Artech House, 1996.

[3] Hein, G. W., Irsigler, M., Avila Rodriguez, J. A., Pany, T., *Performance of Galileo L1 Signal Candidates*, Proc. ENC-GNSS 2004, Rotterdam, May, 2004

[4] *SiRFstarIIA GPS System on Chip*, SiRF Technology, Inc., May, 2005

[5] Garin, L. J., *The "Shaping Correlator", Novel Multipath Mitigation Technique Applicable to GALILEO BOC(1,1) Modulation Waveforms in High Volume Markets*, Proc. ENC-GNSS 2005, Munich, July, 2005

[6] Pany, T., Irsigler, M., Eisfeller, B., *Optimum Coherent Discriminator Based Code Multipath Mitigation By S-Curve Shaping For BOC(N,N) and BPSK Signals*, Proc. ENC-GNSS 2005, Munich, July, 2005

[7] Sleewaegen, J.-M., Boon, F., *Mitigating Short-Delay Multipath: a Promising New Technique*, ION GPS 2001, Salt Lake City, September, 2001

[8] Kovar, P., Vejraska, F., Seidl, L., Kacmarik, P., *Galileo Receiver Core Technologies*, GNSS 2004, Sydney, December, 2004

[9] Glennon, E. P., Dempster, A. G., *A Review of Cross Correlation Mitigation Techniques*, GNSS 2004, Sydney, December, 2004

[10] Dovis, F., Pini, M., Mulassano, P., *Turbo DLL: an Innovative Architecture for Multipath Mitigation in GNSS Receivers*, ION GNSS 2004, Long Beach September, 2004

[11] Babu, R., Wang, J., *Improving the Quality of IMU-Derived Doppler Estimates for Ultra-Tight GPS/INS Integration*, ENC-GNSS 2004, Rotterdam, May, 2004

[12] Keutzer, K., Malik, S., Newton, A. R., *From ASIC to ASIP: The next Design Discontinuity*, ICCD 2002 Proceedings, 2002

[13] Neumann, B., von Sydow, T., Blume, H., Noll, T. G., *Design and quantitative analysis of parametrisable eFPGA-architectures for arithmetic*, Kleinheubacher Tagung, September, 2005

[14] *LISATek Creator's Manual*, CoWare, Oktober 2004