Adv. Radio Sci., 10, 207–213, 2012
www.adv-radio-sci.net/10/207/2012/
doi:10.5194/ars-10-207-2012

Advances in Radio Science

Area- and energy-efficient CORDIC accelerators in deep sub-micron CMOS technologies

U. Vishnoi and T. G. Noll

Chair of Electrical Engineering and Computer Systems, RWTH Aachen University, Aachen, Germany

Correspondence to: U. Vishnoi (vishnoi@eecs.rwth-aachen.de)

Abstract. The COordinate Rotate DIgital Computer (CORDIC) algorithm is a well known versatile approach and is widely applied in today's SoCs for especially but not restricted to digital communications. Dedicated CORDIC blocks can be implemented in deep sub-micron CMOS technologies at very low area and energy costs and are attractive to be used as hardware accelerators for Application Specific Instruction Processors (ASIPs). Thereby, overcoming the well known energy vs. flexibility conflict. Optimizing Global Navigation Satellite System (GNSS) receivers to reduce the hardware complexity is an important research topic at present. In such receivers CORDIC accelerators can be used for digital baseband processing (fixed-point) and in Position-Velocity-Time estimation (floating-point). A micro architecture well suited to such applications is presented. This architecture is parameterized according to the wordlengths as well as the number of iterations and can be easily extended for floating point data format. Moreover, area can be traded for throughput by partially or even fully unrolling the iterations, whereby the degree of pipelining is organized with one CORDIC iteration per cycle. From the architectural description, the macro layout can be generated fully automatically using an in-house datapath generator tool. Since the adders and shifters play an important role in optimizing the CORDIC block, they must be carefully optimized for high area and energy efficiency in the underlying technology. So, for this purpose carry-select adders and logarithmic shifters have been chosen. Device dimensioning was automatically optimized with respect to dynamic and static power, area and performance using the in-house tool. The fully sequential CORDIC block for fixed-point digital baseband processing features a wordlength of 16 bits, requires 5232 transistors, which is implemented in a 40-nm CMOS technology and occupies a silicon area of 1560 µm^2 only. Maximum clock frequency from circuit simulation of extracted netlist is 768 MHz under typical, and 463 MHz under worst case technology and application corner conditions, respectively. Simulated dynamic power dissipation is 0.24 uW MHz^{-1} at 0.9 V; static power is 38 uW in slow corner, 65 uW in typical corner and 518 uW in fast corner, respectively. The latter can be reduced by 43 % in a 40-nm CMOS technology using 0.5 V reverse-backbias. These features are compared with the results from different design styles as well as with an implementation in 28-nm CMOS technology. It is interesting that in the latter case area scales as expected, but worst case performance and energy do not scale well anymore.

1 Introduction

The CORDIC-approach is a well known versatile algorithm, which can be used to implement a large variety of arithmetic functions being quite complex to implement on standard processor architectures. Consequently, it is widely applied in Digital Signal Processing (DSP) on today's SoCs for especially but not restricted to digital communications. Typical tasks comprise (Numerically Controlled Oscillator) NCO, phase rotating, filtering, modulation and demodulation to name a few. A very attractive application of this is to use dedicated CORDIC macros as accelerators for certain arithmetic functions to Application Specific Instruction Processors (ASIPs). If the CORDIC macro is properly optimized for very low area and energy costs, the overhead becomes very small and by that a remedy to the well known "flexibility vs. energy" conflict can be achieved.

In this contribution, CORDIC macros are considered for the use in highly area- and energy efficient digital GNSS receivers. It will be shown in the later sections of the paper that CORDIC can be used in the baseband signal processing part as well as in the so-called Position Velocity Time (PVT) estimation unit. In the future, further applications of CORDIC building blocks can be expected. For example, it can be used in Singular Value Decomposition- (SVD) based jammer or interferer supression in multisystem GNSS receivers.

In order to cover the different specifications for these GNSS receiver applications and also to ensure re-usability in other application domains, a highly flexible, parameterized CORDIC architecture was elaborated. Thereby, a major goal was to achieve an architecture being well suited for implementation in today's deep sub-micron CMOS technologies.

This paper is divided into 5 sections. A brief description of CORDIC algorithm is presented in Sect. 2. Section 3 refers to the proposed flexible CORDIC architecture template and the building blocks required for its implementation are described in detail. Implementation of an exemplary CORDIC macro and results also are presented in the end of Sect. 3. In Sect. 4, as an exemplary application, a CORDIC macro well-suited as a floating-point co-processor to an ASIP is discussed.

2 CORDIC algorithm

The Coordinate Rotation Digital Computer (CORDIC) algorithm was first introduced by Volder (1959) for the computation of trigonometric functions. Later on, it was generalized to hyperbolic functions, multiplication, division etc. by Walther (1971). The algorithm allows for a hardware efficient implementation of arithmetic and trigonometric functions by rotating a vector in a two-dimensional plane using only shift- and addition/subtraction operations being well suited for VLSI-CMOS realization. Thereby, a given input vector $(\boldsymbol{x}_{\text{in}}, \boldsymbol{y}_{\text{in}})$ is rotated in an iterative approach by a series of predetermined angles in each step; the direction of each elementary rotation is determined similarly to non-restoring division.

In more detail, after an initialization

$$x_0 = \boldsymbol{x}_{\text{in}}, y_0 = \boldsymbol{y}_{\text{in}}, z_0 = z_{\text{in}} \tag{1}$$

the vector is iteratively rotated by applying the CORDIC equations

$$\begin{aligned} x_{i+1} &= x_i - m \cdot y_i \cdot d_i \cdot 2^{-i} \\ y_{i+1} &= y_i + x_i \cdot d_i \cdot 2^{-i} \\ z_{i+1} &= z_i - d_i \cdot e_i, \end{aligned} \tag{2}$$

where, $e_i = \tan^{-1}\left(2^{-i}\right)$ is the i-th element of the predefined step angle sequence. The parameter m specifies whether the rotations are performed on a circle ($m = 1$), on a straight line ($m = 0$), or on a hyperbola ($m = -1$). There are two modes of CORDIC operations for each of the above defined rotations: the "rotate mode" and the "vectoring mode". For this application it is sufficient to consider the case of ($m = 1$) with the two possible CORDIC operations. In the rotate mode, the vector $(\boldsymbol{x}_{\text{in}}, \boldsymbol{y}_{\text{in}})$ has to be rotated by a given angle z_{in} and in order to converge z_i to zero the direction d_i of rotation in each individual iteration is determined by the sign of the remaining angle z_i

$$d_i = \text{sgn}(z_i). \tag{3}$$

In the vectoring mode, the vector has to be rotated into the x-axis and therefore, in order to converge y_i to zero the direction d_i of rotation is determined by the sign of the remaining angle z_i

$$d_i = -\text{sgn}(y_i). \tag{4}$$

As Eq. (2) actually perform so-called pseudo-rotations only, at the very end, after n iterations the x- and y-components have to be corrected by a constant K .

$$\begin{aligned} x_{\text{out}} &= K \cdot x_n \\ y_{\text{out}} &= K \cdot y_n \\ z_{\text{out}} &= K \cdot z_n. \end{aligned} \tag{5}$$

This constant is determined by the series of step angles only and especially being independent from the individual directions of rotation d_i. The series of step angles can be optimized by repeating individual elements towards a correction constant K featuring a small number of non-zero digits in canonical signed digit representation easing these final multiplications.

3 CORDIC architecture template

As can be seen from Eq. (2), only additions/subtractions and shift operations are required for CORDIC iterations. In addition to that, registers are required in pipelined CORDIC architectures and multiplexers are required in the feedback time-shared architectures.

Figure 1 shows the block diagram of a fully iterative CORDIC macro performing rotations on a circle ($m = 1$). The input $rot/vect$ is used to switch between rotate and vectoring mode and Initialization is controlled by the input $init$. A single CORDIC stage is multiplexed in time i.e. for each iteration the intermediate results are fed back. The multiplication by 2^{-i} is implemented using shifters which are controlled by the iteration index i. A look-up table (not shown in the figure) feeds the step-angle values e_i, 2's complement subtraction is implemented by using XOR stages and proper LSB 2's complement correction. Final multiplication by the correction constant K, can be implemented by simple shift and add operations (not shown in the Fig. 1).

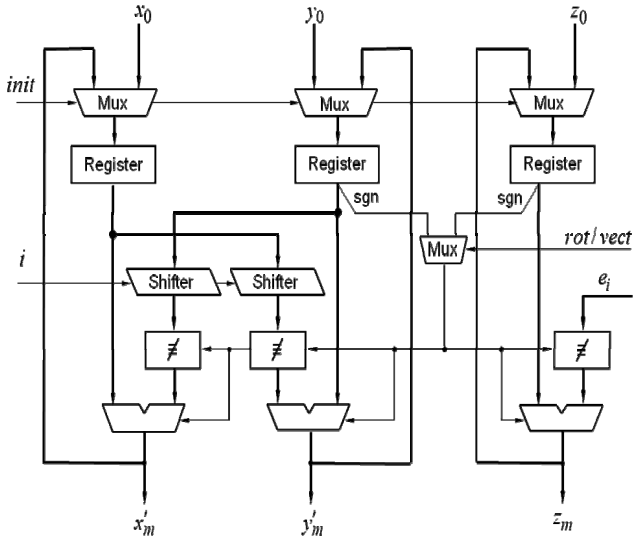

Fig. 1. Basic structure of a fully iterative CORDIC macro.

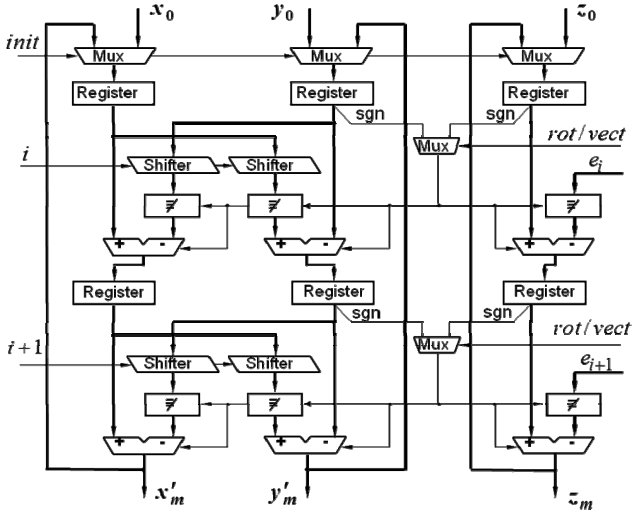

Fig. 2. Partially unrolled CORDIC block diagram (unrolling factor $k = 2$).

As shown in Fig. 2, area can be traded for throughput by partially unrolling the iterative CORDIC structure. Applying two (unrolling factor $k = 2$) pipelined stages increases the area by a factor of about two, but halves the number of clock cycles per CORDIC operation and improves the throughput by a factor of about two. The unrolling can be extended to the case where the unrolling factor k becomes equal to the number of iterations n. With increasing k, the requirements to the programmable shifters reduce. Finally for the fully unrolled structure no shifters are required at all anymore, just the dedicated interconnects are enough. In conclusion, the parameters of this architecture template consist of wordlengths, number of iterations and unrolling factor.

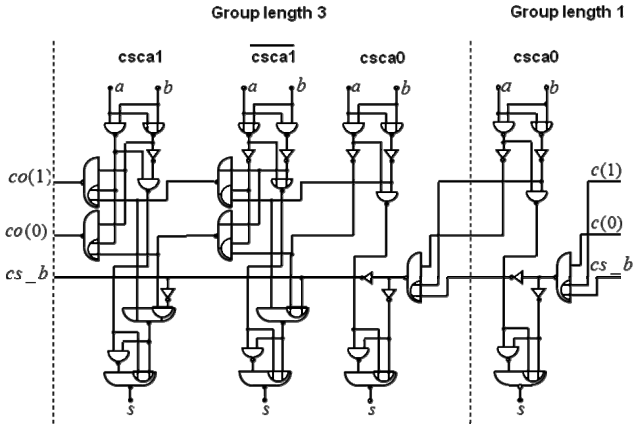

Fig. 3a. The basic cells of carry select adder.

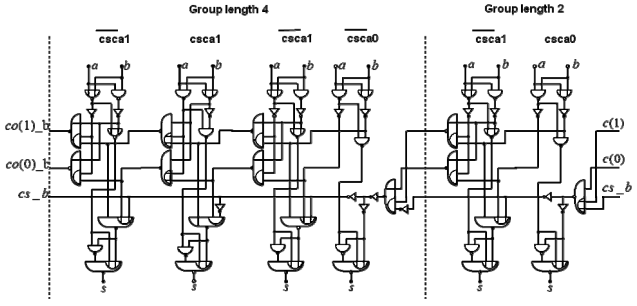

Fig. 3b. The basic cells of carry select adder.

3.1 Building blocks

The major building blocks of the template architecture are the adders and shifters. For the envisaged applications with wordlengths in the range of 16 to 32 bits, carry select adders offer a high area- and energy efficiency (i.e. performance over cost ratio). Figure 3a shows the cell arrangement in the LSBs of carry select adder following the simple arithmetic progression of group lengths 1, 3, 5, 7... and Fig. 3b shows the cell arrangement for the progression of 2, 4, 6, 8... In conclusion, from only four basic cells, delay optimal carry select adders with wordlengths of 9, 12, 16, 20, 25, 30, 36 can be generated and for other wordlengths the MSB group has to be pruned.

The basic cells were implemented in 40-nm CMOS technology; Fig. 3c shows the layouts. To improve the Area-Throughput-Energy (ATE) efficiency, device sizes of the basic cells were optimized automatically by using an in-house tool. All devices belonging to the critical path are carefully optimized for the best possible timing. The tool starts from minimum sized n- and p- devices and in every iteration all the possible permutation of increasing the transistor widths are analyzed. If no further improvement can be achieved the iterations are stopped. Finally from a set of dimensioning candidates the one with lowest ATE complexity was selected. By

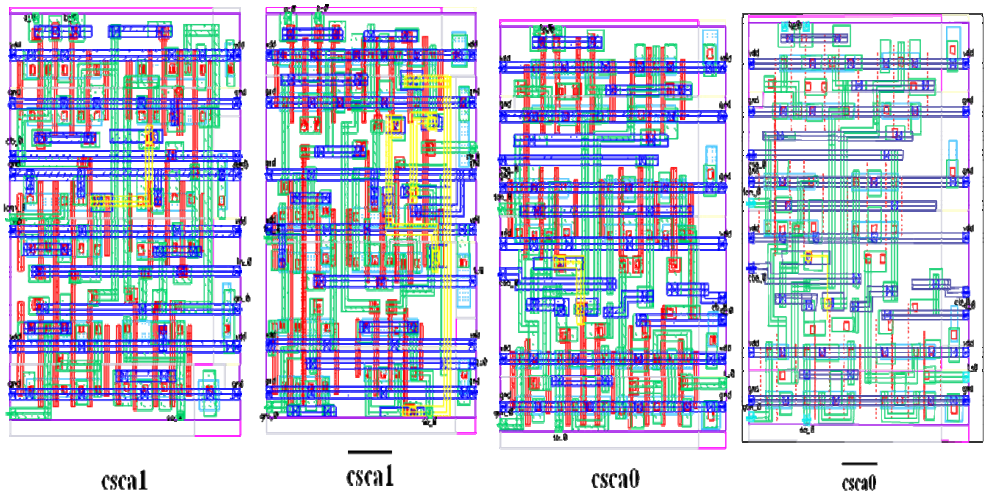

Fig. 3c. Layouts of the basic cells of the carry select adder.

applying this procedure to the adder basic cells, typically a performance improvement of about 25 % was achieved, without sacrificing the efficiency. Moreover, on the positive note, the ATE complexity could even be reduced by typically 5 %.

For the shifter, a logarithmic shifter structure was chosen because it offers high efficiency for the range of above mentioned wordlengths. Only one basic cell namely a one-bit multiplexer had to be implemented in layout. The multiplexer cells as well as the other cells (XOR, register) were optimized in the same way as was described for the carry select adder cells.

3.2 Implementation and results

The CORDIC template architecture was described as a parameterized signal flow graph. With an in-house datapath generator tool, a complete CORDIC macro layout could be generated automatically from only five basic cells for different sets of parameters.

CORDIC macros for two exemplary cases in a digital GNSS receiver were implemented and analyzed in detail.

The first one is used in the baseband processing unit and requires 16 bits wordlength in integer data format and 10 iterations (Kappen, 2011). Figure 4 shows the layout of the CORDIC macro ($k = 1$) without including the Read Only Memory- (ROM) based step angle look-up table in it which can be placed into the blank space. In 40-nm CMOS technology, it contains 5232 transistors in total and occupies a silicon area of 1560 µm^2 only. The maximum clock frequency from extracted netlist simulation is 768 MHz under typical, and 463 MHz under worst case technology and application corner conditions, respectively. Total power dissipation at 500 MHz and 0.9 V supply voltage is 185 uW only.

As already mentioned, one of the architecture parameters is the unrolling factors k. Figure 5 shows the Area-

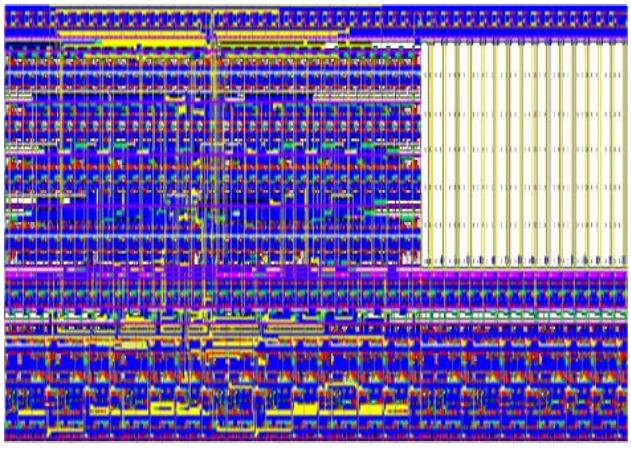

Fig. 4. Layout of CORDIC macro in 40-nm CMOS technology.

Throughput (AT) complexity diagram for CORDIC blocks with $n = 10$ iterations and for different unrolling factors k. For partial unrolling, silicon area increases with a factor of less than two because one level of the shifters can be saved. In the case of fully unrolled macro silicon area is significantly smaller than would be expected from ideal scaling as the shifters can be completely replaced by dedicated interconnects. A fully unrolled CORDIC macro ($k = 10$) allows for a minimum cycle time of 1.2 ns under typical and 2ns under worst case conditions.

The fully iterative macro ($k = 1$) was also extracted and simulated in 28-nm CMOS technology, leading to 56 % of saving in area and a maximum clock frequency is 876 MHz under typical, and 555 MHz under worst case conditions, respectively. Total power dissipation at 500MHz and 0.9 V supply voltage is 134 uW.

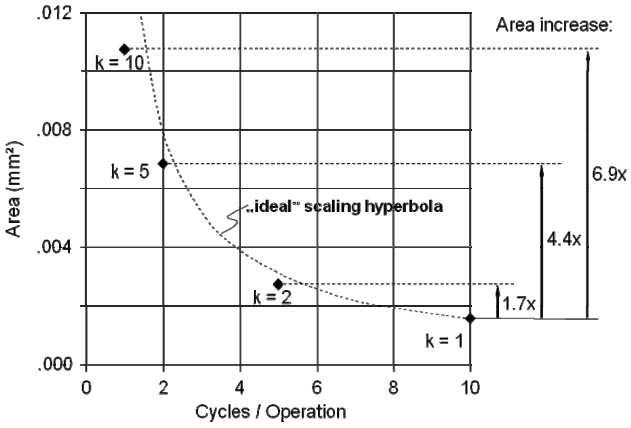

Fig. 5. AT complexity diagram for different unrolling factors k.

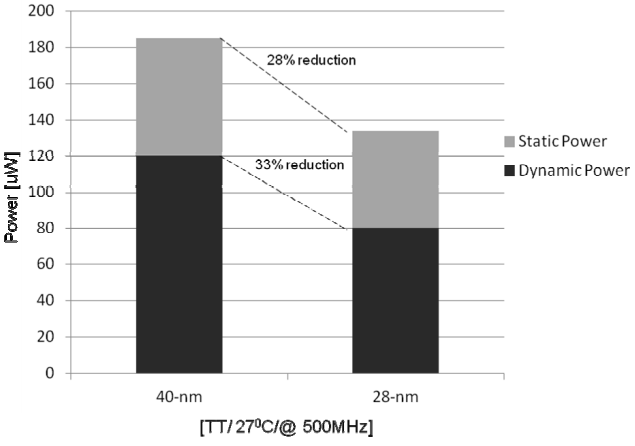

Fig. 6a. Power breakdown at 500 MHz [TT/27 °C/500 MHz].

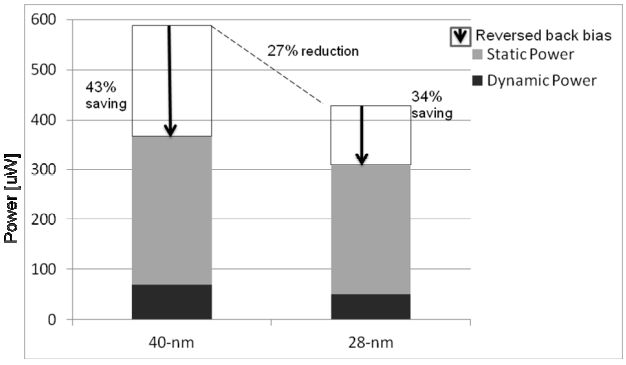

Fig. 6b. CORDIC macro power dissipation.

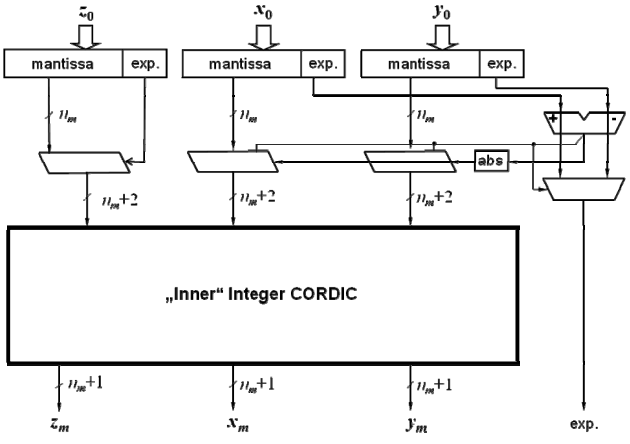

Fig. 7a. Schematic of the Pre-processing for a floating point CORDIC.

Figure 6a shows the power breakdown under typical conditions for 40- and 28-nm CMOS technologies. In the fast corner at 125 °C as shown in Fig. 6b static power dominates by far for both technologies.

And if a reverse back bias voltage of 0.5 V is applied, 43 % of the total power is saved in 40 nm and 34 % of the total power is saved in 28-nm CMOS technology.

4 CORDIC macro as an ASIP co-processor

The second CORDIC macro is used as a co-processor to an ASIP, which in the digital GNSS receiver is mainly used to perform the PVT estimation (Kappen, 2011). In this case floating point data format with 28 bits mantissa wordlength and a number of $n = 30$ iterations is required. As long as rotations are performed only on circles, integer CORDIC macros can be used for floating point data format too with only some simple pre-processing as shown in Fig. 7a. As the length of the vector is kept constant during rotation (except for the correction factor) an initial normalization of x_0 and y_0 has to be performed so that the exponents get aligned. This ensures that in all the successive iterations the mantissa words can be added/subtracted without further normalization.

So, after this pre-processing the integer CORDIC can be used, if provided two extra front bits. One extra bit is required because the x- and y-components can be enlarged by a factor of $\sqrt{2}$ during rotation and another extra bit is required because the correction by K is done at the very end. This pre-processing can be implemented on dedicated hardware. In case the CORDIC is used as a co-processor to an ASIP an attractive alternative is to get the normalization done by the ASIP itself. Same is true for the post-processing, i.e. the multiplication of the output values with the correction factor K.

Figure 7b shows the block diagram of the coupling between the ASIP and the co-processor. In principle, the co-processor can be tightly or loosely coupled to the ASIP. For the application in PVT estimation a tightly coupled co-processor is the preferred alternative.

Table 1 contains the features of the GNSS-ASIP, of non-optimized synthesized floating point CORDIC and of the

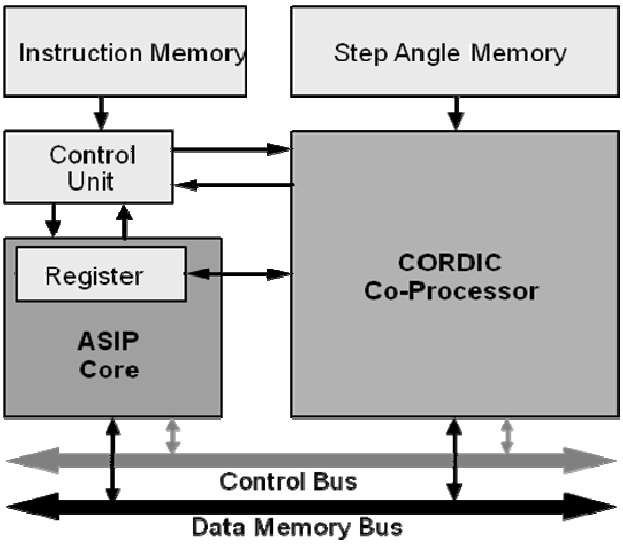

Fig. 7b. Block diagram of the ASIP/CORDIC co-processor structure.

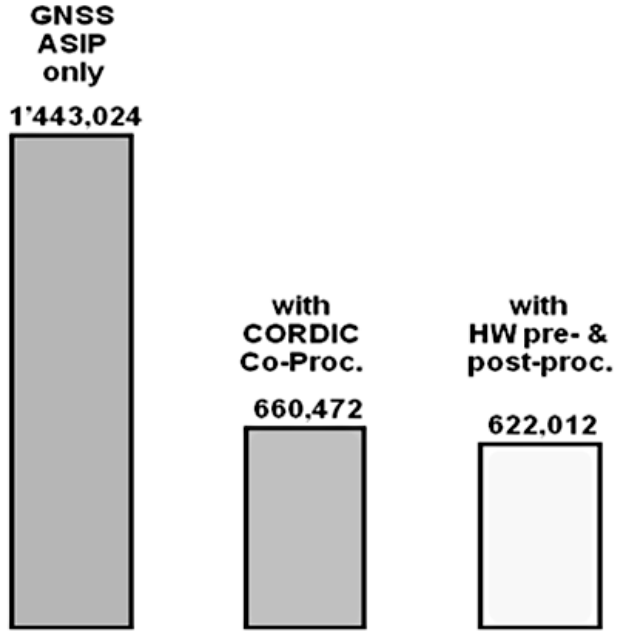

Fig. 7c. Clock cycle per PVT calculation (scenario with 7 satellites).

implementation presented here. The features for the ASIP as well as for the synthesized CORDIC were derived from a 90-nm CMOS implementation (Kappen, 2011) and properly scaled down to 40-nm CMOS technology. As can be seen from the table, the non-optimized CORDIC operated significantly slower than the ASIP itself and the optimized CORDIC would be well-suited to accelerate a faster version of the ASIP. As an alternative with a modified coupling the co-processor could be operated at 500 MHz while the ASIP is operated at about 100 MHz. Moreover, the silicon area and energy per clock cycle of the optimized CORDIC macro are improved by orders of magnitude in comparison to the non-optimized one and are absolutely negligible in comparison to the ASIP.

Table 1. GNSS-ASIP/Co-Processor features (40 nm).

	Synthesized GNSS ASIP IP	Not optimized Synthesized FP CORDIC	30-bit Optimized CORDIC
Silicon area	0.02 mm^2	0.013 mm^2	0.0033 mm^2
Max. clock Frequency (typ.)	96 MHz	35 MHz	> 500 MHz
Energy per clock cycle (typ. at fmax)	12 pJ	9 pJ	0.78 fJ

Figure 7c shows a comparison of the number of clock cycles required per PVT calculation (for constellation with 7 satellites). The ASIP without co-processor requires 1 443 024 clock cycles. The ASIP with co-processor and with the pre- and post-processing also being done by the ASIP requires 660 472 clock cycles. If the floating point pre- and post-processing would be realized in hardware too, then the number of clock cycles would be further reduced to 622 012. Clearly, the most attractive compromise is to let the pre- and post- processing be done by the ASIP which shows an improvement by a factor of 2.18 in cycle count per PVT estimation.

This improvement can even be increased by letting the ASIP continue to execute instructions concurrently to the multi-cycle operations of the co-processor, i.e. the ASIP is not halted during the CORDIC executions.

5 Conclusions

A micro architecture for highly flexible, scalable, area and energy efficient CORDIC macro being well-suited for implementation in deep sub-micron CMOS technologies has been elaborated. Exemplary macros were designed in 40-nm and 28-nm CMOS technologies for application in a digital GNSS receiver. These macros feature very attractive performance, area and power figures. As to be expected for those technologies worst case power dissipation is dominated by static power. But even in the most advanced 28-nm CMOS technology static power dissipation can be significantly reduced by reverse back biasing.

It has also been shown that coupling of the CORDIC macros to an ASIP usage as a co-processor is a very attractive application. For the case of requirement of floating point data format it has been shown that mapping the proper pre-

and post-processing to the ASIP and applying only an integer CORDIC macro yields an attractive solution.

Acknowledgements. The authors thank their colleague Götz Kappen for discussion and for features of the GNSS ASIP.

References

Kappen, G.: Konzeption und Entwurf Flächen- und Energie-effizienter digitaler GNSS-Empfänger, PhD thesis, RWTH Aachen University, June, 2011.

Kappen, G., Kurz, L., Priebe, O., and Noll, T. G.: Design Space Exploration for an ASIP/Co-Processor Architecture used in GNSS Receivers, J. Signal Process. Sys., 1, 41–51, 2008.

Volder, J. E.: The CORDIC trigonometric Computing technique, IRE Trans. Electronic Computers, EC-8, 330–334, September, 1959.

Walther, J. S.: A unified algorithm for elementary functions, AFIPS Spring Joint Computer Conference, 38, 379–85, 1971.