Hardware implementation of an electrostatic MEMS-actuator linearization

authored by **F. Mair, M. Egretzberger, A. Kugi**

and published in *Microsystem Technologies*.

Cite this article as:

F. Mair, M. Egretzberger, A. Kugi, "Hardware implementation of an electrostatic mems-actuator linearization", *Microsystem Technologies*, 18, 7, 955–963, 2012. DOI: 10.1007/s00542-011-1420-x

BibTex entry:

```
@article{acinpaper,
  author = {Mair, F. and Egretzberger, M. and Kugi, A.},
  title = {Hardware implementation of an electrostatic MEMS-actuator linearization},
  journal = {Microsystem Technologies},
  year = {2012},
  volume = {18},
  number = {7},
  pages = {955--963},
  doi = {10.1007/s00542-011-1420-x},
  url = {http://link.springer.com/article/10.1007%2Fs00542-011-1420-x}
}
```

Link to original paper:

http://dx.doi.org/10.1007/s00542-011-1420-x
http://link.springer.com/article/10.1007/s00542-011-1420-x

Microsystem Technologies manuscript No.
(will be inserted by the editor)

Hardware Implementation of an Electrostatic MEMS-Actuator Linearization

F. Mair · M. Egretzberger · A. Kugi

Received: date / Accepted: date

Abstract In this paper, an electrostatic actuator linearization will be introduced, which is based on an existing hardware-efficient iterative square root algorithm. The algorithm is solely based on add and shift operations while just needing $n/2$ iterations for an n bit wide input signal. As a practical example, the nonlinear input transformation will be utilized for the design of the primary mode controller of a capacitive MEMS gyroscope and an implementation of the algorithm will be instantiated in the Verilog hardware description language. Furthermore, an implementation of an improved version of the algorithm will be given reducing the number of needed iterations to $n/2-1$ for an n bit wide input signal while just requiring an acceptable additional amount of hardware resources. Finally, measurement results will validate the feasibility of the presented control concept and its hardware implementation.

Keywords capacitive MEMS gyroscope · electrostatic actuator · actuator linearization · iterative square root algorithm · control of primary oscillator · Verilog HDL

1 INTRODUCTION

The electrostatic actuation principle is the most common way (Bell et al, 2005) to generate actuation forces in micro-electro-mechanical systems (MEMS). The fact that the required electrodes can be manufactured within well established production processes led to its successful application in many mass products like accelerometers, gyroscopes, optical mirrors and many more. A negative aspect, though, that is inherent to all voltage controlled electrostatic actuators is that the generated force is proportional to the square of the applied voltage. However, complex arithmetic calculations are not feasible in MEMS applications as the demands on high sampling rates and low latency, combined with the pricing pressure of high volume production require an efficient hardware implementation of the control loops. Therefore, in most state-of-the-art applications linear control concepts are utilized and the intrinsic limitations in either the range of operation or the lack of performance are accepted. For advanced nonlinear control concepts (Egretzberger, 2010) of capacitive gyroscopes, which result in a noticeable increase of the closed-loop performance, a nonlinear input transformation is indispensable.
In this contribution, an electrostatic actuator linearization will be introduced, which is based on an existing efficient iterative square root algorithm for unsigned integer numbers (Li and Chu, 1996). The advantage of the presented algorithm is that it is solely based on add and shift operations. In comparison to other well established calculation procedures, this iterative algorithm leads to a significant reduction of the required hardware resources, especially as no multipliers are utilized, while just needing $n/2$ iterations for an n bit wide input signal. Furthermore, an implementation of the algorithm in the Verilog hardware description language as well as the implementation of an improved version of the algorithm will be given and the corresponding hardware consumption of both versions of the algorithm will be instantiated for a *Xilinx Virtex 5* Field Programmable Gate Array (FPGA). As a practical exam-

Florian Mair
Automation and Control Institute,
Complex Dynamical Systems Group,
Vienna University of Technology,
Gußhausstr. 27–29, 1040 Vienna, Austria
Tel.: +43-1-58801-37622
Fax: +43-1-58801-37699
E-mail: mair@acin.tuwien.ac.at

ple, the nonlinear input transformation will be utilized for the design of the primary mode controller of a capacitive MEMS gyroscope and measurement results will validate the feasibility of the presented control concept and its hardware implementation.
The paper is organized as follows. Sec. 2 discusses the working principle of a capacitive gyroscope. In Sec. 3 the derivation of the equations of motion are outlined and the nonlinear input transformation is described. The subsequent Sec. 4 gives a short survey on common approaches to calculate the square root operator and presents the implementation of an existing efficient iterative algorithm as well as the implementation of an improved version of the algorithm. The corresponding measurement results are illustrated in Sec. 5 and finally, the contribution is concluded with a short summary.

2 A CAPACITIVE GYROSCOPE

The MEMS element considered as a practical example for the nonlinear input transformation within this article is a capacitive gyroscope (Günthner, 2006), illustrated in Fig. 1, which can measure an externally applied angular rate Ω_y about the sensitive y_0-axis by exploiting the *Coriolis effect*. The capacitive gyroscope is an etched silicon device that uses voltage controlled capacitive actuators (Seeger and Boser, 2003) and capacitive sensors for the excitation and read-out of the in-plane drive and the out-of-plane sense oscillators. As depicted in Fig. 1, the gyroscope comprises a fixed frame, which is rigidly attached to the package of the sensor, and two rigid movable frames, which are connected with the fixed frame via elastic beam elements. Furthermore, four rigid paddles are flexibly connected to the movable frames via elastic torsion beams. Both the comb and the parallel plate actuators and sensors consist of two, in the undeformed configuration parallel, electrodes. One of the electrodes is rigidly mounted on the package of the sensor and the other is rigidly attached to one of the movable frames or paddles resulting in parallel plate capacitors with a capacitance depending on the deflection of the movable structure and therefore allowing for the utilization as actuators and sensors. Applying a harmonic voltage to the drive electrodes results in a harmonic, antisymmetric oscillation of the movable frames and the paddles in x_0-direction (so-called primary mode), as depicted in Fig. 2(a). Due to the high stiffness of silicon, the low actuation forces and the weak damping, the primary oscillator can only achieve reasonable amplitudes if it is excited near its resonance frequency. On the occurrence of an externally applied angular rate Ω_y, the *Coriolis force* couples to the velocity of the primary mode oscillation, resulting in a harmonic out-of-plane oscillation of the paddles and the movable frames in z_0-direction (so-called secondary mode), as illustrated in Fig. 2(b). The harmonic change in capacitance of the electrostatic sensors, i.e. the comb electrodes for the primary mode and the parallel plate electrodes for the secondary mode, is converted to a proportional voltage output signal by appropriately designed charge and differential amplifier circuits.

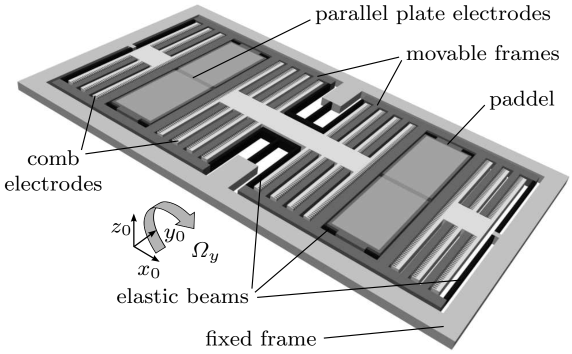
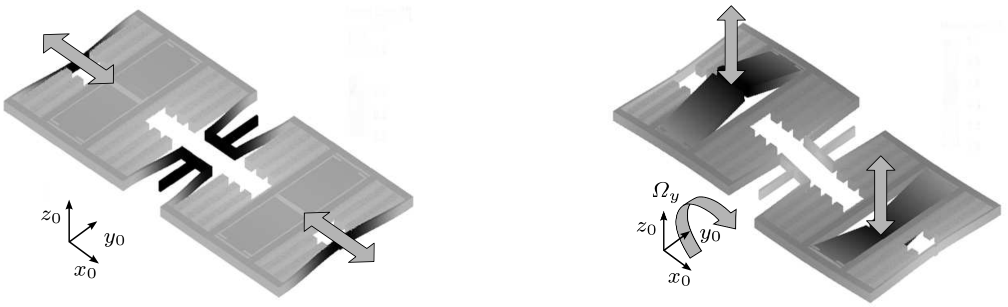


Fig. 1 Capacitive gyroscope assembly.

3 MATHEMATICAL MODEL

As outlined in the previous section, the capacitive gyroscope comprises several rigid and elastic bodies as well as various electrostatic actuators and sensors. Therefore, the derivation of a mathematical model, suitable for a systematic controller design, is rather laborious. For this reason, specialized tools (Mair et al, 2009) have been developed, which automatically derive the analytical equations of motion from CAD input data by dividing the device under consideration into so-called functional components. For each functional component the energy or coenergy is calculated and *Lagrange's formalism* is applied to calculate the corresponding system of differential equations in symbolic form. The thus obtained equations of motion of the device usually cover a dynamic range far beyond the interest of the controller design. Hence, it is reasonable to perform a modal transformation of the system, resulting in a semi-symbolic mathematical model. In a next step, a modal order reduction can be carried out to provide a mathematical model with reduced complexity including solely the relevant dynamics for the controller design (Egretzberger, 2010; Mair et al, 2009). Typically, the first primary and secondary differential modes as well as the first primary and secondary common modes are considered. However,

Post-print version of the article: F. Mair, M. Egretzberger, A. Kugi, "Hardware implementation of an electrostatic mems-actuator linearization", *Microsystem Technologies*, 18, 7, 955–963, 2012. DOI: 10.1007/s00542-011-1420-x



(a) Primary mode

(b) Secondary mode

Fig. 2 Capacitive gyroscope (a) primary mode and (b) secondary mode.

to demonstrate the idea of the nonlinear input transformation it is feasible to keep the equations as simple as possible and therefore we restrict ourselves to the relevant equation of motion of the primary oscillator. Assuming linear damping, linear stiffness and neglecting the effects of the coupling of the secondary oscillator, the equation of motion of the primary oscillator is given by the simple second order differential equation

$$m_1 \ddot{\mathrm{q}}_1 + d_1 \dot{\mathrm{q}}_1 + k_1 \mathrm{q}_1 = \tau_1(u_1) \tag{1a}$$

$$y_1 = c_1 \mathrm{q}_1 \tag{1b}$$

with the modal degree-of-freedom q_1, the modal mass m_1, the modal damping coefficient d_1, the modal stiffness coefficient k_1, the system output y_1, the output coefficient c_1 and the nonlinear input force

$$\tau_1(u_1) = b_1 u_1^2 \tag{2}$$

with the input coefficient b_1. Both, the input coefficient b_1 and the output coefficient c_1 are defined by the geometric design of the drive actuators and sensors. Performing a nonlinear input transformation for (1a) of the form

$$u_1 = \sqrt{\tilde{u}_1} \quad \text{with} \quad 0 \leq \tilde{u}_1 \tag{3}$$

yields a simple linear second order differential equation with the new input $\tilde{u}_1$. As described in the previous Sec. 2, the functional principle of the capacitive gyroscope requires that the excitation of the primary mode is close to the resonance frequency of the primary oscillator. However, for the controller design the slow dynamics (envelope) of the primary mode signal is relevant and not the fast harmonic carrier signal itself. Therefore, it is reasonable to introduce a so-called envelope model (Egretzberger and Kugi, 2010). Let us assume a harmonic excitation of the primary oscillator by means of an input signal of the form

$$\tilde{u}_1 = |\tilde{U}_1| + \tilde{U}_1 \cos(\omega t) \tag{4}$$

with the amplitude $\tilde{U}_1$ and the excitation frequency ω. If the motion of the primary mode is approximated in the form

$$\mathrm{q}_1(t) = \mathrm{Q}_{1,S} \sin(\omega t) + \mathrm{Q}_{1,C} \cos(\omega t) \tag{5}$$

with the Fourier coefficients $\mathrm{Q}_{1,S}$ and $\mathrm{Q}_{1,C}$, then the simplified envelope model of the primary mode is given by (Egretzberger et al, 2010)

$$\begin{bmatrix} \dot{\mathrm{Q}}_{1,S} \\ \dot{\mathrm{Q}}_{1,C} \end{bmatrix} = \begin{bmatrix} -\alpha_1 & \omega - \omega_1 \\ \omega_1 - \omega & -\alpha_1 \end{bmatrix} \begin{bmatrix} \mathrm{Q}_{1,S} \\ \mathrm{Q}_{1,C} \end{bmatrix} - \begin{bmatrix} \beta_1 \\ 0 \end{bmatrix} \tilde{U}_1 \tag{6a}$$

$$\begin{bmatrix} \mathrm{Y}_{1,S} \\ \mathrm{Y}_{1,C} \end{bmatrix} = \begin{bmatrix} \gamma_1 & 0 \\ 0 & \gamma_1 \end{bmatrix} \begin{bmatrix} \mathrm{Q}_{1,S} \\ \mathrm{Q}_{1,C} \end{bmatrix} \tag{6b}$$

with the Fourier coefficients of the output signal $\mathrm{Y}_{1,S}$ and $\mathrm{Y}_{1,C}$. The damping coefficient α_1 and the resonance frequency ω_1 read as

$$\alpha_1 = \frac{1}{2} \frac{d_1}{m_1}, \quad \omega_1 = \frac{1}{2} \frac{1}{m_1} \sqrt{4 k_1 m_1 - d_1^2} \tag{7}$$

and the input and output coefficients β_1 and γ_1 are given by

$$\beta_1 = \frac{1}{2} \frac{b_1}{m_1 \omega_1}, \quad \gamma_1 = c_1. \tag{8}$$

Applying an output transformation (Egretzberger et al, 2010)

$$\mathrm{Y}_{1,A} = \sqrt{\mathrm{Y}_{1,S}^2 + \mathrm{Y}_{1,C}^2}, \quad \mathrm{Y}_{1,\phi} = \arctan\left(\frac{\mathrm{Y}_{1,S}}{\mathrm{Y}_{1,C}}\right) \tag{9}$$

with the amplitude $\mathrm{Y}_{1,A}$ and the phase $\mathrm{Y}_{1,\phi}$ to the envelope model (6) and calculating the steady state

$$\mathrm{Y}_{1,A} = \frac{\beta_1 \gamma_1 \tilde{U}_1}{\sqrt{\alpha_1^2 + (\omega - \omega_1)^2}}, \quad \mathrm{Y}_{1,\phi} = \arctan\left(\frac{\alpha_1}{\omega_1 - \omega}\right) \tag{10}$$

allows for a straightforward specification of the necessary control tasks. As can be inferred from the steady state (10), the amplitude $\mathrm{Y}_{1,A}$ of the primary mode gets maximal for a fixed input amplitude $\tilde{U}_1$, if the oscillator is excited at its resonance frequency, i.e. $\omega = \omega_1$, while maintaining a phase $\mathrm{Y}_{1,\phi} = -\pi/2$. Therefore, for a proper operation of the capacitive gyroscope the amplitude, frequency and phase of the primary oscillator have to be controlled simultaneously.

At this point it is worth mentioning that it is possible to achieve a linear envelope behavior of the primary oscillator (Egretzberger et al, 2010) without applying the proposed nonlinear input transformation (3), resulting in the shortcoming that the primary oscillator is additionally excited by a signal part with double the excitation frequency. Furthermore, as the constant or the harmonic component of the excitation signal has to be set to a constant value, the remaining control input is constrained by this value. Moreover, for advanced nonlinear control concepts (Egretzberger, 2010) of the secondary oscillator, which result in a noticeable increase of the closed-loop performance, a nonlinear input transformation in the form of a square root operator is indispensable.

4 ARITHMETIC ALGORITHM

Depending on the demands on throughput, latency and hardware-efficiency as well as on the available resources and the interaction with other arithmetic operations, different approaches for the realization of the square root operator have been proposed in the literature. A complete survey on existing algorithms is far beyond the scope of this contribution, nonetheless, care has been taken to include the most relevant ones, at least for the application under consideration. In the following, it is assumed that the radicand is a positive real number and that it is given in a conventional number representation, e.g. floating or fixed point, and conventional number system, e.g. normalized binary floating point or radix 2. Conventional floating point numbers are considered in the following short survey, because the time-consuming part of the root calculation of a floating point number has to be performed on the mantissa, for which the same algorithms can be applied as for fixed point numbers. However, the necessary adjustment of the exponent and the inevitable postnormalization steps between the mantissa and the exponent add considerable additional complexity to the circuitry.

All algorithms familiar to the authors, calculate the square root of a number in an iterative manner. Therefore, if a single clock cycle calculation is mandatory, a function approximation (Meyer-Baese, 2007) might be the only solution, resulting in a very high hardware effort as, depending on the order of the approximation, many parallel single-cycle multipliers are needed. Furthermore, the accuracy of the result is, due to obvious reasons, limited. The iterative algorithms can be separated into two categories (Soderquist and Leeser, 1997), i.e. multiplicative algorithms and subtractive algorithms. The multiplicative algorithms derive the square root by an iterative refinement of an initial guess and divide the calculation into a series of multiplications, additions and shift operations. The most common representatives are the *Newton Raphson* method (Hennessy and Patterson, 2007) and the *Goldschmidt's* algorithm (Hennessy and Patterson, 2007). Both possess a quadratic convergence and, an adequate seed generator presumed, approximate the square root in a few iteration cycles very well. Though, due to the multiplication operation at each iteration step, a low or single cycle multiplier is inevitable for an effective implementation and hence, the hardware effort is high. The subtractive algorithms on the other hand are solely based on add, shift and relational operations, which can be implemented in hardware very efficiently. Several well known algorithms exist that need $n/2$ bits for an n bit wide radicand, however, they differ substantially in their hardware consumption. Among these algorithms are the iterative restoring and non-restoring square root extraction (Koren, 2002), the algorithm based on the bisection method (Tommiska, 2000; Dijkstra, 1976) as well as the very hardware effective non-restoring square root algorithm (Li and Chu, 1996), which can be implemented solely based on add and shift operations. An improvement in regards to the needed number of iterations can be achieved by the SRT algorithm (Koren, 2002), which, depending on the applied input data, calculates the result in equal or less than $n/2$ steps. As data-dependent delays are not feasible in practical implementations, the data-independent high radix SRT algorithms (Koren, 2002) are of greater practical importance. For the current application under consideration, the choice of an applicable algorithm is based on several considerations. First, the excitation voltage (3), which is generated by the digital signal processing unit of the capacitive gyroscope, has to be converted from the digital to the analog domain by a Digital to Analog Converter (DAC) comprising a fixed bit width and a defined output voltage range. Therefore, we can restrict ourselves to algorithms suitable for fixed point numbers only. Second, the sample rate of the DAC is usually an integer multiple smaller than the internal clock signal, allowing for the execution of intermediate calculation steps. This brings along that the number of required iterations is not the primary concern. The last point that has to be considered is the additional latency which is introduced by the algorithm. To resolve

Post-print version of the article: F. Mair, M. Egretzberger, A. Kugi, “Hardware implementation of an electrostatic mems-actuator linearization”, *Microsystem Technologies*, 18, 7, 955–963, 2012. DOI: 10.1007/s00542-011-1420-x
The content of this post-print version is identical to the published paper but without the publisher’s final layout or copy editing.

this issue, it is feasible to examine the input signal (4) in more detail. The output signal of the amplitude controller $\tilde{U}_1$ is a slow signal, which gets modulated with the fast harmonic carrier signal $\cos(\omega t)$. The phase signal $\phi = \omega t$ is generated by the internal frequency and phase controller and can be shifted by adding an offset ϕ_a. Therefore, the latency of the algorithm can be easily compensated in the form

$$\tilde{u}_{1,a} = |\tilde{U}_1| + \tilde{U}_1 \cos(\omega t + \phi_a) \quad \text{with} \quad \phi_a = \frac{2\pi m \omega}{\omega_s}\frac{3}{2}, \tag{11}$$

with the new excitation signal $\tilde{u}_{1,a}$, the adjustment angle ϕ_a, the excitation frequency ω, the internal clock sampling frequency ω_s and the required number of calculation steps m. The factor $3/2$ centers the resulting excitation voltage $u_{1,a}$ in regard to the ideal excitation voltage u_1. As all concerns regarding the calculation efficiency can be easily resolved, the most hardware effective algorithm (Li and Chu, 1996) was chosen. An implementation of the iterative square root algorithm in the Verilog hardware description language (HDL) is given in Listing 1 and an implementation of an improved version of the algorithm is instantiated in Listing 2. The improved version decreases the required numbers of iterations by calculating the first and last iteration step at the same time. This approach requires an additional register to store two different intermediate results of the root variable at the same time resulting in a slightly higher amount of hardware resources. However, buffering the output of the module by an additional register prevents the appearance of intermediate results at the output and furthermore leads to a better software design hierarchy. The resulting hardware consumptions of both versions of the algorithm on a *Xilinx Virtex 5* FPGA for a 12 bit wide root, i.e. 24 bit wide radicand, are depicted in Tab. 1. As can be seen, the improved version requires an even smaller amount of look-up-tables as the original algorithm while just needing an adequate additional number of D-flip-flops. It has to be mentioned at this point that the hardware consumption may depend on the utilized synthesis and place and route tools as well as on the selected synthesis and optimization preferences. To ease the comparison for the interested reader, the standard Xilinx toolchain (Xilinx, 2010) with the standard preferences have been used for the presented case. As a practical example, the excitation signals $\tilde{u}_1$ and $\tilde{u}_{1,a}$, with $\tilde{U}_1 = 1/2$ and the resulting excitation voltages $u_1 = \sqrt{\tilde{u}_1}$ and $u_{1,a} = \sqrt{\tilde{u}_{1,a}}$ are illustrated in Fig. 3(a) and Fig. 3(b), respectively. Henceforth, $\bar{t}$ refers to the normalized time t.

Algorithm	original	improved
radicand (bit)	24	24
required iterations	12	11
look-up-tables	67	60
D-flip-flops	30	41
carry-chains	2	2

Table 1 Hardware consumption of the iterative square root algorithm on a *Xilinx Virtex 5* FPGA.

5 MEASUREMENT RESULTS

To verify the nonlinear input transformation (3) introduced in Sec. 3, an amplitude, phase and frequency controller for the primary oscillator as well as the square root algorithm of Sec. 4 have been implemented on a development board, consisting of a *Xilinx Virtex 5* FPGA and additional analog circuitry for the actuation and read out of a prototype gyroscope. The measurement results of the static behavior of the closed-loop amplitude controller at different reference points are illustrated in Fig. 4. The deflection signal and the control input are normalized in the form $\bar{\mathrm{Q}}_{1,S} = \mathrm{Q}_{1,S}/\mathrm{Q}_{1,S,d}$ and $\bar{U}_1 = \tilde{U}_1/\tilde{U}_{1,d}$ with the nominal point of operation $\mathrm{Q}_{1,S,d}$ and the associated control input $\tilde{U}_{1,d}$, respectively. As depicted in Fig. 4, the applied normalized control input signal $\bar{U}_1$ and the normalized primary mode deflection $\bar{\mathrm{Q}}_{1,S}$ show a perfect linear relation. Furthermore, to confirm the linear dynamic behavior of the amplitude controller over the full control input range, i.e. during acceleration as well as deceleration, the tracking behavior of the closed-loop amplitude controller with respect to a stepwise defined reference signal $\bar{\mathrm{Q}}_{1,r}$ and the applied normalized control input $\bar{U}_1$ are illustrated in Fig. 5(a) and Fig. 5(b), respectively.

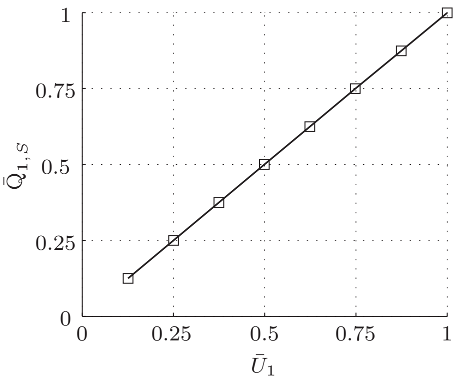
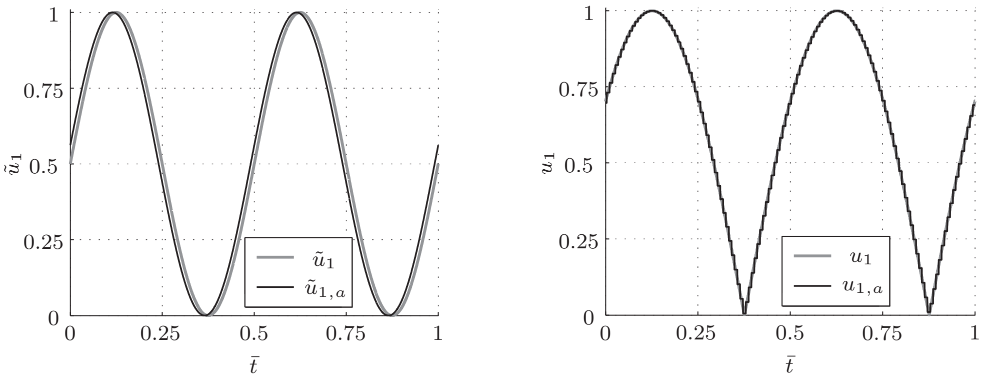
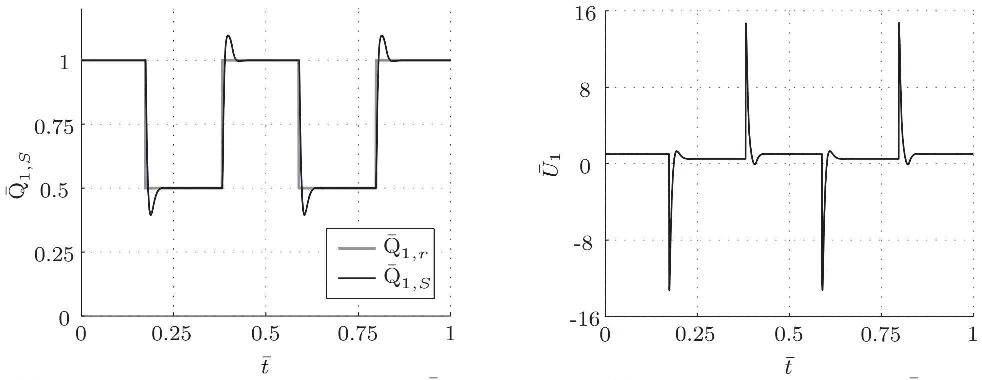


Fig. 4 Static behavior of the closed-loop amplitude controller at different normalized reference points $\bar{\mathrm{Q}}_{1,S}$ and the applied normalized control input $\bar{U}_1$.



(a) Input signals $\tilde{u}_1$ and $\tilde{u}_{1,a}$ (b) Excitation voltages u_1 and $u_{1,a}$

Fig. 3 (a) Input signals $\tilde{u}_1$ and $\tilde{u}_{1,a}$ and resulting (b) excitation voltages u_1 and $u_{1,a}$.



(a) Normalized primary mode deflection $\bar{Q}_{1,S}$ (b) Normalized control input $\bar{U}_1$

Fig. 5 (a) Dynamic behavior of the closed-loop amplitude controller with respect to a stepwise defined reference signal $\bar{Q}_{1,r}$ and (b) the applied normalized control input $\bar{U}_1$.

The measurement results of both experiments, i.e. the static as well as the dynamic closed-loop behavior of the amplitude controller, validate the effectiveness of the nonlinear input transformation, its underlying square root algorithm as well as the proposed hardware implementations.

6 SUMMARY AND OUTLOOK

In this paper, an electrostatic actuator linearization is introduced, which is based on an existing efficient iterative square root algorithm for unsigned integer numbers. Furthermore, an implementation of the algorithm in the Verilog hardware description language is given and the corresponding hardware consumption is instantiated for a *Xilinx Virtex 5* Field Programmable Gate Array (FPGA). Finally, as a practical example, the nonlinear input transformation is utilized for the design of the primary mode controller of a capacitive MEMS gyroscope and measurement results validate the feasibility of the presented control concept and its hardware implementation.

APPENDIX

ACKNOWLEDGMENTS

This work was funded by the German BMBF as part of the EURIPIDES project RESTLES (project number 16SV3579).

References

Bell D, Lu T, Fleck N, Spearing S (2005) MEMS actuators and sensors: observations on their performance and selection for purpose. Journal of Micromechanics and Microengineering 15:153–164

Dijkstra EW (1976) A Discipline of Programming. Prentice-Hall, Englewood Cliffs, USA

```
module iter_square_root #(
    // --- Parameters ------------------------------------------------------------
    parameter ROOT_W = 12                       //Width of "root"
    )(
    // --- Inputs ----------------------------------------------------------------
    input clk,
    input reset_N,
    input enable,
    input [2*ROOT_W-1:0] radicand,
    // --- Outputs ---------------------------------------------------------------
    output reg [ROOT_W-1:0] root,
    output valid
    );
    // --- Local Parameters ------------------------------------------------------
    localparam NRSTATES = ROOT_W;               //Number of internal states
    localparam REM_W    = ROOT_W+1;             //Width of "remainder_"
    localparam RAD_W    = ROOT_W*2;             //Width of "radicand"
    // --- Variables -------------------------------------------------------------
    reg signed [REM_W-1:0] remainder_;
    reg signed [CLog2(NRSTATES):0] state_; //CLog2() calculates the ceil of log2()
    // --- Nets and Continuous Assignments ---------------------------------------
    wire [ROOT_W-1:0] root_s1_ = (root <<< 1);
    wire [ROOT_W+1:0] root_s2_ = (root <<< 2);
    wire [1:0] radicand_b_ = radicand >> (state_ <<< 1);
    wire signed [REM_W+1:0] remainder_m_ =
                        $signed({remainder_[REM_W-1:0], radicand_b_[1:0]}) - (root_s2_ | 1);
    wire signed [REM_W+1:0] remainder_p_ =
                        $signed({remainder_[REM_W-1:0], radicand_b_[1:0]}) + (root_s2_ | 3);
    always @(posedge clk, negedge reset_N) begin
        if (!reset_N) begin
            root        <= 0;
            remainder_  <= 0;
            state_      <= -1;
        end else begin
            if (enable) begin
                if (state_ < 0) begin
                    state_      <= NRSTATES-2;
                    remainder_ <= $signed({{(REM_W-2){1'd0}},radicand[(RAD_W-1)-:2]}) - 1;
                    root        <= {{(ROOT_W-1){1'd0}},(radicand[RAD_W-1]|radicand[RAD_W-2])};
                end else begin
                    if (!remainder_[REM_W-1]) begin
                        remainder_ <= remainder_m_[REM_W-1:0];
                        root        <= {root_s1_[ROOT_W-1:1],~remainder_m_[REM_W+1]};
                    end else begin
                        remainder_ <= remainder_p_[REM_W-1:0];
                        root        <= {root_s1_[ROOT_W-1:1],~remainder_p_[REM_W+1]};
                    end
                    state_ <= state_ - 1;
                end
            end
        end
    end
    assign valid = (state_ < 0) ? 1:0;
endmodule
```

Listing 1 Verilog-HDL implementation of the iterative square root algorithm.

Egretzberger M (2010) Mathematical Modeling and Control of Micro Electromechanical Gyroscopes. Shaker, Aachen, Germany

Egretzberger M, Kugi A (2010) A dynamical envelope model for vibratory gyroscopes. Microsystem Technologies 16:777–786

Egretzberger M, Mair F, Kugi A (2010) Control of vibratory MEMS gyroscopes based on envelope models. In: IFAC Symposium on Mechatronic Systems, Cambridge, USA, pp 441–446

Günthner S (2006) Entwurf und Charakterisierung von mikromechanischen Drehratensensoren in Silizium. Shaker, Aachen, Germany

Hennessy JL, Patterson DA (2007) Computer Architecture, A Quantitative Approach, 4th edn. M. Kaufmann, San Francisco, USA

Koren I (2002) Computer Arithmetic Algorithms, 2nd edn. A K Peters, Natick, USA

Li Y, Chu W (1996) A new non-restoring square root algorithm and its VLSI implementations. In: Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors, Austin, USA, pp 538–544

Mair F, Egretzberger M, Kugi A (2009) A tool for the automatic modeling of capacitive MEMS gyroscopes. In: Proceedings of the 6th Vienna International Conference on Mathematical Modelling, Vienna, Austria, pp 2228–2235

Meyer-Baese U (2007) Digital Signal Processing with Field Programmable Gate Arrays, 3rd edn. Springer,

```
module iter_square_root_improved #(
    // --- Parameters ------------------------------------------------------------
    parameter ROOT_W = 12                     //Width of "root"
    )(
    // --- Inputs ----------------------------------------------------------------
    input clk,
    input reset_N,
    input enable,
    input [2*ROOT_W-1:0] radicand,
    // --- Outputs ---------------------------------------------------------------
    output reg [ROOT_W-1:0] root,
    output valid
    );
    // --- Local Parameters ------------------------------------------------------
    localparam NRSTATES = ROOT_W;             //Number of internal states
    localparam REM_W    = ROOT_W+1;           //Width of "remainder_"
    localparam RAD_W    = ROOT_W*2;           //Width of "radicand"
    // --- Variables -------------------------------------------------------------
    reg [ROOT_W-1:0] root_;
    reg signed [REM_W-1:0] remainder_;
    reg [CLog2(NRSTATES)-1:0] state_; //CLog2() calculates the ceil of log2()
    // --- Nets and Continuous Assignments ---------------------------------------
    wire [ROOT_W-1:0] root_s1_ = (root_ <<< 1);
    wire [ROOT_W+1:0] root_s2_ = (root_ <<< 2);
    wire [1:0] radicand_b_ = radicand >> (2*state_);
    wire signed [REM_W+1:0] remainder_m_ =
                         $signed({remainder_[REM_W-1:0], radicand_b_[1:0]}) - (root_s2_ | 1);
    wire signed [REM_W+1:0] remainder_p_ =
                         $signed({remainder_[REM_W-1:0], radicand_b_[1:0]}) + (root_s2_ | 3);
    always @(posedge clk, negedge reset_N) begin
        if (!reset_N) begin
            root        <= 0;
            root_       <= 0;
            remainder_  <= 0;
            state_      <= 0;
        end else begin
            if (enable) begin
                if (state_ == 0) begin
                    if (!remainder_[REM_W-1]) begin
                        root <= {root_s1_[ROOT_W-1:1],~remainder_m_[REM_W+1]};
                    end else begin
                        root <= {root_s1_[ROOT_W-1:1],~remainder_p_[REM_W+1]};
                    end
                    remainder_ <= $signed({{(REM_W-2){~(radicand[RAD_W-1]|radicand[RAD_W-2])}},
                                  ~(radicand[RAD_W-1]^radicand[RAD_W-2]),~radicand[RAD_W-2]});
                    root_      <= {{(ROOT_W-1){1'd0}},(radicand[RAD_W-1]|radicand[RAD_W-2])};
                    state_     <= NRSTATES-2;
                end else begin
                    if (!remainder_[REM_W-1]) begin
                        remainder_ <= remainder_m_[REM_W-1:0];
                        root_      <= {root_s1_[ROOT_W-1:1],~remainder_m_[REM_W+1]};
                    end else begin
                        remainder_ <= remainder_p_[REM_W-1:0];
                        root_      <= {root_s1_[ROOT_W-1:1],~remainder_p_[REM_W+1]};
                    end
                    state_ <= state_ - 1;
                end
            end
        end
    end
    assign valid = (state_ == 0) ? 1:0;
endmodule
```

Listing 2 Verilog-HDL implementation of the improved iterative square root algorithm.

Berlin Heidelberg, Germany

Seeger JI, Boser BE (2003) Charge control of parallel-plate, electrostatic actuators and the tip-in instability. Journal of Microelectromechanical Systems 12:656–671

Soderquist P, Leeser M (1997) Division and square root: choosing the right implementation. Micro, IEEE 17:56–66

Tommiska MT (2000) Area-efficient implementation of a fast square root algorithm. In: Proceedings of the 2000 Third IEEE International Caracas Conference on Devices, Circuits and Systems, Cancun, Mexico, pp 18/1–18/4

Xilinx (2010) PlanAhead User Guide 12.3. Xilinx, Inc., URL www.xilinx.com