Bit-Serial CORDIC: Architecture and Implementation Improvements

Johan Löfgren and Peter Nilsson
Dept. of Electrical and Information Technology, Box 118, Lund University, Sweden
Email: {Johan.Lofgren, Peter.Nilsson}@eit.lth.se

***Abstract*—This paper presents a new and improved bit-serial CORDIC architecture. A detailed description of the bit-serial implementation and its Control Unit is presented. It is shown that the improvement is due to a reduction of registers in the implementation and is made possible by ensuring that the angular path is calculated prior to the corresponding vector paths. In addition, the improved architecture is implemented in VHDL and synthesized for a UMC 130 nm technology. With the chosen parameters, a word length of 12 bits and 8 stages in the CORDIC, it is shown that the improved architecture is 20 % smaller and consumes 26 % less power.**

I. Introduction

In modern hardware implementations, the leakage power has become an increasing problem. This is due to the lowered threshold voltages, V_Ts, of the MOS transistors, which will lead to higher cut-off currents. When the dimensions of the technologies become even smaller, it is predicted that the leakage power will be the main power consumer in the hardware [1].

There are different methods and ideas to reduce the leakage power. One way to go is to run the whole circuit in the transistor sub-threshold region [2]. This will yield a better relation between dynamic and static power, but the throughput of the circuit will decrease dramatically, since the transistor switching time is much greater in this region. It also reduces the noise margins since it requires low voltages. Another possible solution to the problem is to power gate the design, i.e., turning of the power for parts of a circuit that is not currently used. If allowed by throughput constraints, it is also possible to use a high-V_T process. A higher V_T reduces leakage, but will also slow down the circuit, albeit not as much as will be the case in the sub-threshold domain, as discussed above [3].

A further possibility is to change the architecture of the implementation and run the data in bit-serial instead of bit-parallel order, that is, operating at one bit at a time, instead of a full word [4]–[6]. It may seem that the bit-serial implementation would be much slower than the bit-parallel but this is not necessarily the case, since it is possible to shorten the critical path considerably using a bit-serial design. A potential drawback with the bit-serial implementation style is that it may lead to an excessive amount of registers and that it thereby will increase the dynamic power consumption. The number of registers needed, depends very much on the algorithms being implemented. In some cases the number of extra register will be higher, in some lower.

In this paper the bit-serial implementation of a COordinate Rotation DIgital Computer (CORDIC) [7], for calculating sine and cosine values, is investigated and implemented. The bit-serial implementation of the CORDIC algorithm often leads to a great need for registers, since each following step is dependent on the sign bit of the preceding, and the sign bit is the last bit calculated in using bit-serial logic. In this paper a novel approach is introduced that reduces the number of needed registers by letting dependent data paths having a later starting time.

A. Previous Works and Motivation

Many papers describes different usage of the CORDIC [7], [8] and bit-serial implementations [9], [10]. There are bit-serial implementations of the CORDIC algorithm presented in literature [10]. However, no implementations focusing on the reduction of registers have been found in the literature. Partially this may reflect that often bit-serial logic is implemented on FPGAs and it is thus not made with the current power focus. Instead most of the implemented architectures are cyclic to save area [9], [10]. In today's technologies the area is often not the major concern. This opens for new solutions on using unrolled architectures [8]. In this context the present work helps by presenting results that improve the bit-serial CORDIC architectures.

II. The CORDIC Algorithm

The CORDIC algorithm is used in different situations and for different purposes. The common, basic idea between the different usages is that through a series of additions and subtractions, it is possible to approximate vector rotations in the complex plane. This can be used in e.g. the calculation of trigonometric or hyperbolic values [9].

The CORDIC works in two different modes; Rotation Mode and Vector Mode. In Rotation Mode the algorithm rotates a vector with a given angle, and in Vector Mode the angle between a given vector and the x-axis is calculated. The algorithm is multiplication-free and thus well suited for hardware implementation.

In this paper, the CORDIC is used for the calculation of the trigonometric functions sine and cosine of an input angle. For that purpose, the CORDIC is working in Rotation Mode, with the input vector being known in advanced, and found to be $x_0 = 1/R$ and $y_0 = 0$, where R is a radius correction factor, that will depend on the number of stages of the CORDIC. The vector of each new stage is then found as

$$\begin{aligned} x_{n+1} &= x_n \pm \tfrac{y_n}{2^n} \\ y_{n+1} &= y_n \mp \tfrac{x_n}{2^n} \end{aligned} \tag{1}$$

with S being the stage number, and the sign of the operation depending of the sign of the angle addition/subtraction.

Further, the input angle is limited to values between 0 and $\pi/2$ (90°). This is not a problem when calculating trigonometric functions, since all angles can easily be moved or mirrored into the first quadrant.

III. Hardware Architecture

The CORDIC consists of an angular path, that handles the operation on the angle, and a vector path for calculations on the x and y values. An unrolled architecture is examined in this paper. This leads to larger hardware but a higher throughput. Since the input angle can be limited to the first quadrant and that the trigonometric functions ensure $y_0 = 0$, the first stage of the calculations can be expanded as

$$\begin{aligned} x_1 &= x_0 - \tfrac{0}{1} = x_0 \\ y_1 &= y_0 + \tfrac{x_0}{1} = x_0 \end{aligned} \tag{2}$$

It is thus possible to remove the first x and y adders in the unrolled chain without any additional hardware cost.

As stated in section II, the input angle, α_{in}, is between 0 and $\pi/2 < 1.75$, which means that if at least two fractional bits are used, it is sufficient to have one integer bit to span the full input range. In addition, to be able to keep the same number representation throughout the whole chain, also negative numbers has to be handled, requiring an extra sign/integer bit.

Also in the x and y streams two integer bits, including the sign bit, are needed. The sign bit is needed since both x and y can take negative values (close to angles of 0 and $\pi/2$ respectively). The need for the extra integer bit is not obvious at first, but due to quantization, it is possible to overflow and end up at values of $x, y \geq 1$ in certain stages of the CORDIC. To safely handle this, an extra integer bit is needed.

A. Original Bit-Serial Architecture

In Fig. 1 the original bit-serial architecture of the CORDIC is shown. The figure shows an architecture with 4 stages and a word length of 8 bits. As shown, the three paths, the angular and the two vector paths are different but highly dependent data paths.

The A/S units are one bit add/subtract units. In order to handle both addition and subtraction with the same hardware adder, there is an initial stage with a conditional inverter for one of the inputs for each A/S unit. To perform a subtraction the control bit of the adder, connected to the A/S unit is set to '1'. This inverts the second operand and sets the input sign bit to '1'. With this construct, no specific subtraction units are needed. As can be seen in the figure, the control bit is the sign bit of the previous stage angle operation. Thus angle calculation of the previous stage must be completed before the addition or subtraction can begin.

The constant angles, α_0 to α_{N-2}, with N being the number of stages, are hard coded in the design. They are stored as negative values and fed to the A/S units of the angular path, bit by bit, through the MUXes. It is noted that the final stage in the CORDIC require no calculation in the angular path, since the value is not used in any later stage and is therefore not calculated.

In addition, a number of one bit registers, the D units, are needed to store the bits between the operations. The registers are connected in series, to create shift registers or First-In-First-Out (FIFO) queues. To divide by two in a bit-serial circuit, the non-divided data has to be delayed one clock cycle (CC). A number of the delay elements found in the circuit are there to enable the division. Other delay elements are needed to ensure proper timing of the signals in the angular and the vector paths. Partially this comes from the need to calculate the angular path sign bit prior to the calculation of add/subtract operation in the vector paths of the following stage.

In addition a number of bit keeper units, or K units, are needed, to keep the control bit/sign bit for a full cycle. The K units sample the data at certain instances and then keep that data for a number of CCs matching the word length in the design. The K units are, as seen in the figure, used for two different tasks. One task is to keep the sign bit of the angular path to control the A/S units. The other task is to keep the sign bit of divided input words. In bit-parallel hardware, division by two is performed by right shifting the data word, or simple rewiring of the bit lines. In bit-serial arithmetic the division is performed by dropping the initial bit. To still match bits correctly, non-divided values may need to be delayed to allow the non-needed bits to be dropped. In addition, the dropping of bits means that the sign bit of the divided word has to be repeated in order to keep the same word length. The K register is thus keeping the sign bit and the MUXes are controlled to emit the stored sign bit when appropriate.

B. Improved Bit-Serial Architecture

A major drawback of the presented bit-serial design is the many, long shift registers. These add to both the dynamic and static power dissipation of the circuit. Looking at the implementation, it is noted that many of the registers are present only to ensure that the vector paths are executed after the previous stage of the angular path. Thus, if the angular path calculations where ready earlier, a number of registers could be avoided. This would lead to an improvement in both area and power figures, since the removed registers would reduce the switching activity.

Fig. 2 shows this improved architecture. Removing the register changes the timing of the circuit. In order for this to work, the angular path calculations must begin earlier than the corresponding vector path calculations. Also, to not increase the latency or reduce the throughput of the design, the vector path calculations should start so that the final add/subtract operation, starts exactly when the final angular operation is finished and the sign bit of it is available. Thus it is important

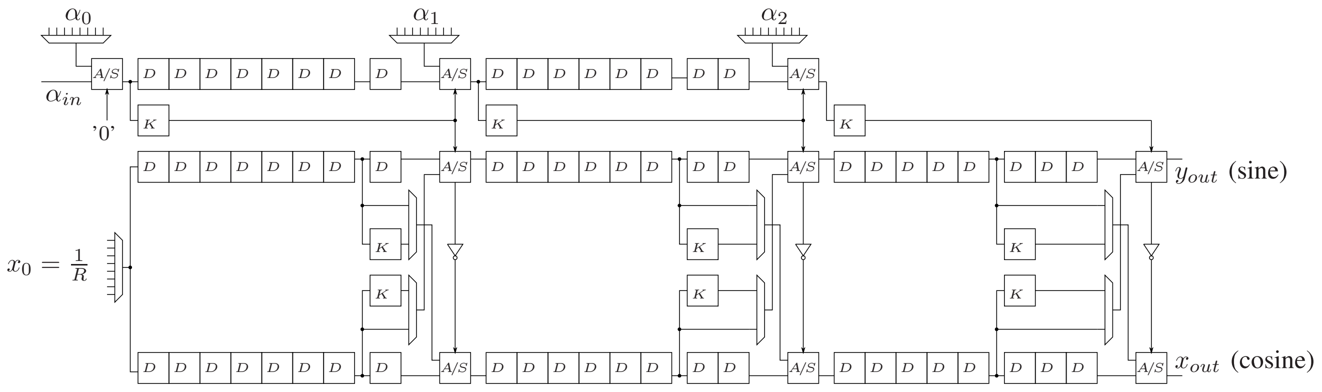

Fig. 1. Bit-Serial Architecture; 4 stages, 8 bit words

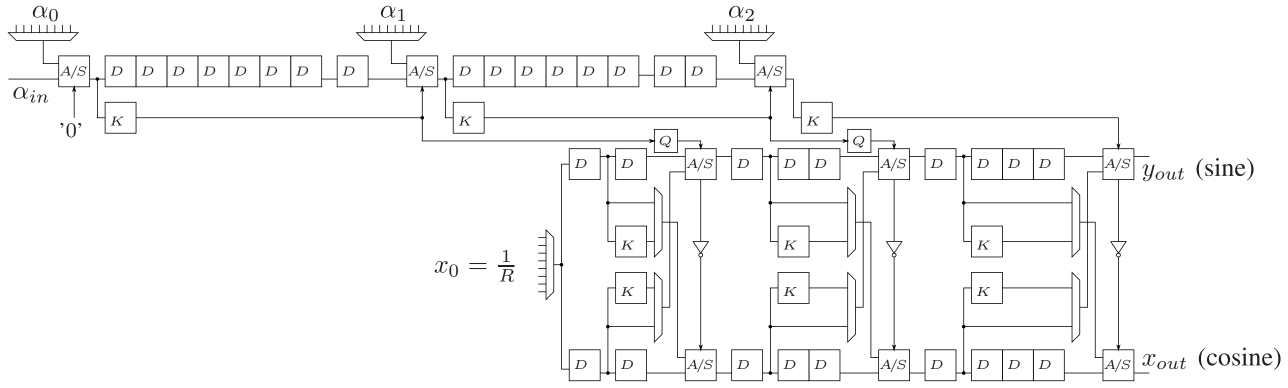

Fig. 2. Improved Bit-Serial Architecture; 4 stages, 8 bit words

to delay the start of the vector path calculation until sufficiently many angular values have been calculated.

Assuming a stream of input angles, so that all stages are active in parallel, the current stage will have the data ready one CC earlier than the following stage, since the next stage contains one more register to allow the division. Thus each stage will be timed differently and will also need to keep the sign bits from the angular paths for some additional time. Therefore the Q registers are introduced. The different Q units need to store the data bits for a different number of CCs. If the number of CCs is greater than the word length, this is achieved by an internal shift register based on K units.

This of course adds to the logic and number of registers of design. It is thus important that this overhead is not greater than the savings from the reduction of registers in the vector paths.

C. Control Unit

A bit-serial implementation has an inherent need for a Control Unit. It is important to reset a bit-serial A/S unit before starting a word operation. This is because the bit-serial units need to keep a state, i.e., the carry bit. That means that the controller somehow must implement a counter, counting up to the word length of the data, to ensure that a new reset signal is emitted before each new word operation is initiated. At the same time, it is important that the critical path is not within

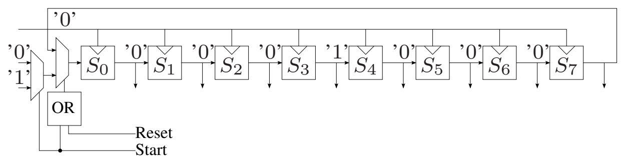

Fig. 3. Bit-Serial Controller in state S_3; 8 bit words

the control unit, to allow the design to run at its maximum clock frequency. This prevents the usage of a counter using an adder, since this could not be bit-serial (because then another controller would be needed to control that one) and not be bit-parallel since that would give a longer critical path.

These considerations lead to a controller implemented as a ring buffer, as shown in Fig. 3. In reset, the controller is filled with '0' bits, where after a single '1' is injected at the start. This '1' will propagate through the ring buffer and will mark the current state. In addition to control the reset signal of the A/S units, the controller will also control the MUXes in the vector path, to emit the stored sign bit, as described above. The different stages will need to use different control states for this.

D. Critical Path

Ideally, the critical path of the bit-serial design should be through a full adder only, to allow maximum speed. In the

TABLE I
SYNTHESIS RESULTS ORIGINAL BIT-SERIAL DESIGN

Critical Path	**Combinational**	**Non-comb**	**Total area**
1.47 ns	3,063 μm^2	8,292 μm^2	11,355 μm^2
2.00 ns	2,207 μm^2	8,165 μm^2	10,372 μm^2
3.27 ns	2,062 μm^2	8,197 μm^2	10,259 μm^2

current design, the critical path starts from the controller, and then goes through the input MUXes that chose whether to use the kept sign bit in the K units or the output from the previous stage. After that follows the A/S unit, which internally contains a conditional inverter, implemented as an XOR gate, and a full adder. In addition there is an extra set-up time on the input of the following D register. The critical path is the same for both the original and the improved design.

The critical path is independent of the number of stages or the word length of the adder, since only one single bit in a single stage is treated at a time. The bit-serial does not produce a new data word every CC, but needs as many CCs as the word length to finish. Thus, the throughput will drop with an increased word length even though the maximum frequency of operation stays the same. In contrast, the bit-parallel design will continue to produce a new output for each CC also with an increased word length, but the maximum operation frequency will drop, due to the increased critical path in the adder/subtracter. Thus the throughput should drop similarly in a bit-parallel design.

IV. IMPLEMENTATION RESULTS

The parameters chosen in the following comparisons are a word length of 12 bits and CORDICs of 8 stages. The precision is increased with the number of stages, but also the area and the power consumption of the circuit will go up.

The VHDL code is synthesized for the UMC 130 nm process, using high V_T standard cells. This will lead to slower logic but also reduced leakage power. Different constraints have been put on the designs, in order to to find both the smallest and the fastest solutions.

A. Speed and Area

Bit-parallel implementations contain long chains of combinational logic. This gives the synthesizing tool great freedom when optimizing the design. Thus, it is possible to get widely varying areas and critical paths based on the constraints. However, for bit-serial implementations there are few degrees of freedom for the synthesize tool, since there are only a few combinational gates between the registers. This means that there is a little spread in both time and area between the smallest and the largest implementation. Table I shows the critical path and the area of the original bit-serial design, whereas the numbers for the improved design are shown in table II.

Comparing the two designs it is seen that the critical paths, and thereby the maximum operating frequencies, of the two designs does not differ much. However, the area of the improved design is 20 % lower (when the critical path is set to 2.00 ns).

TABLE II
SYNTHESIS RESULTS IMPROVED BIT-SERIAL DESIGN

Critical Path	**Combinational**	**Non-comb**	**Total area**
1.51 ns	3,073 μm^2	6,099 μm^2	9,172 μm^2
2.00 ns	2,162 μm^2	6,086 μm^2	8,248 μm^2
2.57 ns	2,065 μm^2	6,052 μm^2	8,118 μm^2

TABLE III
SIMULATED POWER

Design	**Clock**	**Reg.**	**Comb.**	**Tot. Power**
Original	207 μW	130 μW	52 μW	389 μW
Improved	151 μW	84 μW	51 μW	287 μW

B. Power Estimation

The power consumption of the two designs has also been simulated, using *PrimeTime* from *Synopsys*. Table III shows the power numbers for the two different implementations, synthesized for a critical path of 2.00 ns, running at 100 MHz.

V. CONCLUSION

In this paper, a new and improved bit-serial CORDIC architecture is presented. It is shown how it is possible to reduce the number of registers by calculating the angular path prior to the vector paths in the CORDIC. Some extra registers and extra logic are needed to store the sign bits of the angular calculations. However, this is substantially less than the area reduction in the vector paths. For the chosen implementation with a word length of 12 and 8 stages, the improved architecture is 20 % smaller and consumes 26 % less power.

REFERENCES

[1] (2009) IEEE ITRS technology roadmap. [Online]. Available: http://public.itrs.net

[2] B. C. Paul, A. Raychowdhury, and K. Roy, "Device optimization for digital subthreshold logic operation," *Electron Devices, IEEE Transactions on*, vol. 52, no. 2, Feb 2005.

[3] A. Agarwal, S. Mukhopadhyay, A. Raychowdhury, K. Roy, and C. H. Kim, "Leakage power analysis and reduction for nanoscale circuits," *IEEE Micro*, Mar 2006.

[4] R. Lyon, "Two's complement pipeline multipliers," *Communications, IEEE Transactions on*, Apr 1976.

[5] L. Wanhammar, *DSP Integrated Ciruits*. Academic Press, 1999.

[6] P. Nilsson, "Architectures and arithmetic for low static power consumption in nanoscale CMOS," *Journal of VLSI Design*, 2009.

[7] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Transactions on Electronic Computers*, 1959.

[8] P. Nilsson, "Complexity reductions in unrolled CORDIC architectures," *Proceedings of the IEEE 14th International Conference on Electronics, Circuits and Systems (ICECS 2009)*, 2009.

[9] R. G. Harber, J. Li, X. Hu, and S. C. Bass, "Bit-serial cordic circuits for use in a VLSI silicon compiler," *ISCAS 1989. IEEE International Symposium on*, 1989.

[10] S. Vadlamani and W. Mahmoud, "Comparison of CORDIC algorithm implementations on FPGA families," *Proceedings of the Thirty-Fourth Southeastern*, 2002.