Journal of VLSI Signal Processing 32, 207–222, 2002

Evaluation of CORDIC Algorithms for FPGA Design

JAVIER VALLS
Department of Ingenieria Electronica, Universidad Politecnica de Valencia, 46730 Grao de Gandia, Valencia, Spain

MARTIN KUHLMANN AND KESHAB K. PARHI
Broadcom Corporation, 16215 Alton Pkwy, Irvine, CA 92619, USA

Received April 19, 2000; Revised November 8, 2000; Accepted January 8, 2001

Abstract. This paper presents a study of the suitability for FPGA design of full custom based CORDIC implementations. Since all these methods are based on redundant arithmetic, the FPGA implementation of the required operators to perform the different CORDIC methods has been evaluated. Efficient mappings on FPGA have been performed leading to the fastest implementations. It is concluded that the redundant arithmetic operators require a 4 to 5 times larger area than the conventional architecture and the speed advantages of the full custom design has been lost. That is due to the longer routing delays caused by the increase of the fan-out and the number of nets. Therefore, the redundant arithmetic based CORDIC methods are not suitable for FPGA implementation, and the conventional two's complement architecture leads to the best performance.

Keywords: CORDIC, FPGA, Two's complement, redundant arithmetic

1. Introduction

The high capability and performance that FPGAs have achieved in last years allow them to accelerate DSP tasks. FPGA devices have been used for implementing Custom DSPs from the beginning of the past decade [1–4]. The FPGA devices have benefized from the improvements in VLSI deep sub-micron technology, leading to higher speed and capability as well as low power consumption.

CORDIC (COordinate Rotation DIgital Computer) [5, 6] is an iterative algorithm for the calculation of the rotation of a two-dimensional vector, in linear, circular and hyperbolic coordinate systems, using only add and shift operations. Its current application are in the field of digital signal processing, image processing, filtering, matrix algebra, etc. [7].

Early FPGAs were not able to implement a parallel CORDIC algorithm due to the limited chip-size and the impossibility of routing the hard-wired shifters [8]. Consequently, it has been performed in several FPGA-based DSP applications as an iterative structure [9–12].

The enhancement of CORDIC algorithm for VLSI design has been an active research topic in last decades [13–26]. However, all these techniques have not been evaluated for FPGA design. This paper emphasizes the realization of these VLSI algorithms on FPGA. These methods have been evaluated and compared with the conventional way to implement CORDIC algorithm.

This paper is organized as follows. In Section 2 the CORDIC algorithm is presented and the problems of its implementation are commented. In Section 3 metrics to compare the different FPGA implementations are outlined. In Section 4 the CORDIC operators required in the different methods are evaluated. The conventional CORDIC implementation is studied in Section 5 and Section 6 is dedicated to the redundant arithmetic based CORDIC methods. Finally, Section 7 concludes the paper.

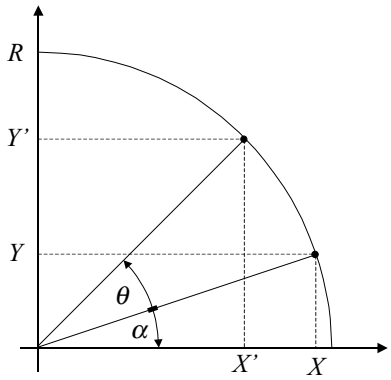

Figure 1. The rotation and vectoring mode of the CORDIC algorithm.

2. CORDIC Algorithm

CORDIC is an iterative algorithm for the calculation of the rotation of a two-dimensional vector, in linear, circular and hyperbolic coordinate systems, using only add and shift operations. It consists of two operating modes, the rotation mode (RM) and the vectoring mode (VM). In the rotation mode a vector (X, Y) is rotated by an angle θ to obtain a new vector (X', Y'). In every micro-rotation i, fixed angles of the value $\arctan(2^{-i})$ which are stored in a ROM are subtracted or added from/to the remainder angle θ_i, so that the remainder angle approaches to zero (Fig. 1). In the vectoring mode, the length R and the angle α towards the x-axis of a vector (X, Y) are computed. For this purpose, the vector is rotated towards the x-axis so that the y-component approaches to zero. The sum of all angle rotations is equal to the value of α, while the value of the x-component corresponds to the length R of the vector (X, Y). Hence, the mathematical relations to be iterated are shown in (1). By selecting the appropriated value for the parameter m in (1) a different coordinate system can be achieved ($m = 0, 1, -1$ corresponds to linear, circular, and hyperbolic coordinate system, respectively).

$$\begin{aligned} X_{i+1} &= X_i - \sigma_i \cdot 2^{-i \cdot m} \cdot Y_i \\ Y_{i+1} &= Y_i + \sigma_i \cdot 2^{-i \cdot m} \cdot X_i \\ Z_{i+1} &= Z_i - \sigma_i \cdot \theta_i \end{aligned} \tag{1}$$

This paper is focused on circular coordinates systems. However, considering FPGA implementations, linear and hyperbolic coordinates can be treated similarly. The difference among them is the shift factor $2^{-i \cdot m}$: there is no shift operation in linear coordinates, there is a right-shift operation in circular coordinates, and there is a left-shift operation in hyperbolic coordinates. Furthermore, in bit-parallel architectures, which are of interest in this paper, a shift operation is performed by a hard-wiring. Hence, the obtained conclusions may be extended to linear and hyperbolic coordinate systems.

In the RM mode the direction of the micro-rotations σ_i are determined by the sign of the Z_i variable, if sign of Z_i is positive $\sigma_i = 1$ otherwise $\sigma_i = -1$. In VM the decision criteria depends on the sign of the Y variable, if it is positive then $\sigma_i = -1$ else $\sigma_i = 1$.

Each iteration in CORDIC is not a perfect rotation, it is a rotation which modifies the length of the vector in a quantity of $(1 + 2^{-2i})^{1/2}$. Therefore, after N iterations its magnitude has changed by a factor K_N in (2). In order to maintain a constant vector length the obtained result has to be scaled by $1/K_N$. However using consecutive rotations the scale factor can be pre-computed.

$$K_N = \prod_{i=0}^{N} \sqrt{1 + \sigma_i^2 \cdot 2^{-2i}} \tag{2}$$

To satisfy an M-bit precision CORDIC operation, $M + 1$ iterations are needed. Furthermore the length of the data-path to compute the X and Y variables has to be $N = M + 2 + \log_2(M)$ and for the computation of Z only a precision of $M + 1$ is needed in the operations [17].

2.1. Redundant Arithmetic Based CORDIC

The drawbacks of conventional CORDIC implementations, based on ripple carry adders or subtractors, is the internal carry propagation delay. To enhance the performance of CORDIC redundant arithmetic has been proposed. This arithmetic, due to its carry free property, avoids the carry propagation from the LSB to the MSB.

Nevertheless, its use involves several difficulties. It is not possible to detect the sign of a redundant number without inspecting all the digits which requires a propagation from the MSB to the LSB. Hence, the decision criteria is chosen according to several most significant digits. Since not all digits are examined there is the possibility that the sign is not determined. In such a case either the digit set $\{-1, 0, 1\}$ can be chosen leading to a non constant scale factor or an arbitrary rotation has to

be performed. In the first case, the scale factor has to be computed in parallel to the CORDIC iterations while a wrong selected rotation requires a compensation to ensure the convergence.

In Section 6 several of the redundant arithmetic based algorithms which solve the previously described problems are examined from the point of view of the FPGA implementation.

3. Comparison of FPGA Implementations

Xilinx FPGA has a matrix structure in which each element of the matrix is a Configurable Logic Block (CLB). The CLBs are interconnected by the available routing resources and surrounded by programmable input-output blocks (IOBs). The routing resources can be connected by means of programmable interconnect to implement the desired routing. A CLB contains two 4-input Look-Up Tables (LUTs), one 3-input one and two registers, as shown in Fig. 2. It also contains dedicated resources to perform a fast propagation of arithmetic carries between adjacent CLBs. This carry chain is independent of the normal routing resources.

A comparison of full custom designs can be easily performed by evaluating critical path delay (for example, counting the number of full adders in between two registers). However, in FPGA design the performance can change depending on, for example, the degree of area utilization. In FPGA it is not sufficient to count the number of LUTs (Look-up Tables) or CLBs (Configurable Logic Blocks) because the routing are significant component of the total delay. The long routing delays are caused by the following items:

- A 100% connectivity among the resources in the FPGA can not be obtained.
- It is not possible to find an optimum placement and routing for all problems due to the limited resources and the fixed structure of the devices.
- Placement and routing are automatic or semi-automatics processes.

In this paper the throughput is characterized by the propagation delay in the critical path. The equation of this delay will be given and within it not only the fixed delays will be included but also the routing delays. To characterize the latter delays its fan-outs and the number of nets that have the same fan-out will be given. With these two parameters it is not possible to determine the magnitude of the delay, nevertheless they are useful to compare different implementations. Obviously a routing delay with higher fan-out will be slower than other one with lower value. Furthermore, higher the number of nets with the same fan-out, higher the probability of longer routing delays. Table 1 shows the values and descriptions of the symbols used to denote the delays in the critical paths. They correspond to a XC4000XL Xilinx chip with speed degree of 1.

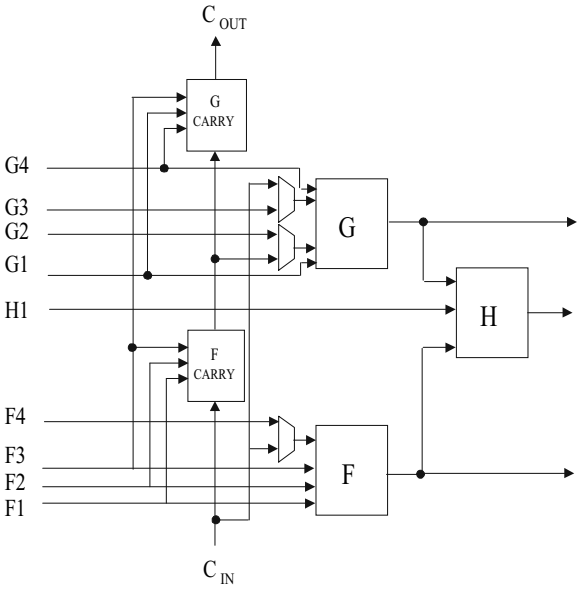

Figure 2. Configurable logic block.

Table 1. Timing parameters.

Symbol	Description	ns
T_{ILO}	F/G inputs to X/Y outputs	1.3
T_{IHO}	F/G inputs via H to X/Y outputs	2.2
T_{OPCY}	Operant inputs ($F1$, $F2$, $G1$, $G4$) to C_{OUT}	2.0
T_{SUM}	C_{IN} through function generators to X/Y outputs	2.4
T_{BYP}	C_{IN} to C_{OUT}, bypass function generators	0.20
T_{NET}	Carry Net Delay	0.25
T_{CKO}	Clock K to Flip-Flop output Q	1.6
T_{ICK}	Setup time before clock K, F/G inputs	0.9
T_{IHCK}	Setup time before clock K, F/G inputs via H	1.7
$T_{\text{net}}^{Xf,Y}$	Routing delay of a net with a fan-out of X, there are Y nets with the same fan-out	—

4. Conventional vs. Redundant Arithmetic on FPGA

The operations involved in CORDIC are addition and subtraction. In this section the FPGA implementation of these operators by using both conventional and redundant arithmetic is studied. Other operators required in redundant arithmetic based CORDIC are also studied.

4.1. Conventional Arithmetic on FPGA

Current FPGAs provide resources to efficiently implement arithmetic operators. Xilinx XC4000 contains dedicated logic for the fast generation of carry signals (Fig. 2). Note that the propagation time between a carry input and a carry output is $T_{\mathrm{BYP}} = 0.20$ ns and the delay between the carry lines of two adjacent CLBs is also fixed, $T_{\mathrm{NET}} = 0.25$ ns.

4.1.1. Adders and Subtractor. The basic cell of the conventional adder and subtractor is the full adder (FA). A full adder is composed by two 3-imput functions; therefore it can be implemented with a single CLB. Nevertheless, if the resources of the FPGA to generate the carry signal are used, a reduction of area and an increase in speed is achieved. In this case two FAs can fit in one CLB. Hence, an N-bits adder or subtractor requires $N/2$ CLBs.

The propagation delay of these circuits is given by:

$$\begin{aligned} T_{\mathrm{p}} &= T_{\mathrm{CKO}} + t_{\mathrm{net}}^{\mathrm{1f},2N} + T_{\mathrm{OPCY}} \\ &\quad + (N-4)/2 \cdot (T_{\mathrm{BYP}} + T_{\mathrm{NET}}) + T_{\mathrm{SUM}} \\ &= \left[6 + t_{\mathrm{net}}^{\mathrm{1f},2N} + (N-4)/2 \cdot 0.45\right] \text{ns}. \end{aligned} \tag{3}$$

The critical path is determined by the propagation time of the ripple carry and the routing delay of one of the least significant inputs of the circuit (each of these two paths have a fan-out 2). A 16-bit adder or subtractor achieves a throughput of 87 MHz in a device with a speed degree of 1 with an occupation lower than 5%.

4.1.2. Adder/Subtractor. An adder/subtractor performs an addition or a subtraction depending on a selection input A/S. This input indicates whether an operand is two's complemented. The add/sub basic cell is decomposed by 2 4-input functions, one to compute the output and the other to transmit the carry. According to this an N-bit adder/subtractor can fit in $(N+1)/2$ CLBs. The additional half CLB is required for adding the LSB 1 in case of the subtraction.

In this circuit the critical path is determined by the ripple carry propagation and the routing delay of the A/S wire. In this case that net has a fan-out of $2N$, which automatically decreases the performance of the circuit. That critical path is expressed as:

$$\begin{aligned} T_{\mathrm{p}} &= T_{\mathrm{CKO}} + t_{\mathrm{net}}^{Nf,1} + T_{\mathrm{ASCY}} \\ &\quad + (N-4)/2 \cdot (T_{\mathrm{BYP}} + T_{\mathrm{NET}}) + T_{\mathrm{SUM}} \\ &= \left[6.5 + t_{\mathrm{net}}^{Nf,1} + (N-4)/2 \cdot 0.45\right] \text{ns}. \end{aligned} \tag{4}$$

The throughput achieved by a 16-bit adder/subtractor fitted in a device with a speed degree of 1 with an area utilization lower than 5% is 70 MHz.

4.2. Radix-2 Redundant Arithmetic on FPGA

Redundant arithmetic is useful to speed-up operations characterized by long propagation delays. Besides that feature, it allows processing the data with most significant digit first (MSDF). This simplifies the implementation of the operators that inherently work in MSDF fashion like division or square root or improved algorithms such as Differential CORDIC [17].

In next sections the FPGA implementation of the operators required to perform the different redundant CORDIC structures are studied. The implementation on FPGA of other operators like multipliers, scaling, squaring and divider can be found in [27].

In radix-2 redundant arithmetic a number X is represented as:

$$X_2 = x_0.x_1x_2\cdots x_{N-1} = \sum_{i=0}^{N-1} x_i 2^{-i}, \tag{5}$$

where each digit, x_i, is from the set $\{-1, 0, 1\}$ according to

$$x_i = x_i^+ - x_i^-, \tag{6}$$

where $x_i^+, x_i^- \in \{0, 1\}$.

4.2.1. Hybrid Addition and Subtraction. Hybrid addition adds an unsigned number, Y, to a redundant one, X_2, resulting in another redundant number, S_2, as indicated in (7). To perform this operation a plus-plus-minus (PPM) cell is needed (Fig. 3(a)), which computes

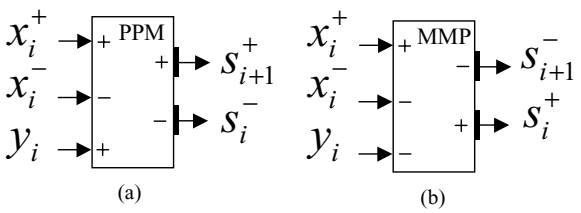

Figure 3. (a) PPM cell, (b) MMP cell.

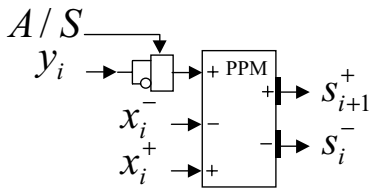

Figure 4. Hybrid add/sub cell.

the equation given in (8).

$$S_2 = X_2 + Y. \tag{7}$$

$$x_i^+ - x_i^- + y_i = 2s_{i+1}^+ - s_i^-. \tag{8}$$

The hybrid subtraction performs the subtraction between a redundant number and an unsigned number as indicated in (9). The basic cell needed is the minus-minus-plus (Fig. 3(b)) which computes the Eq. (10).

$$S_2 = X_2 - Y. \tag{9}$$

$$x_i^+ - x_i^- - y_i = -2s_{i+1}^- + s_i^+. \tag{10}$$

PPM and MMP cells require 1 CLB each to be implemented. Therefore, the occupation of an N-bit hybrid redundant adder or subtractor is N CLBs, twice larger than the conventional ones.

The critical path in these circuits is composed by a fixed delay, which does not depend on the size of the operators, and a routing delay with a probability of being longer than that in the conventional adder or subtractor (11). An N-bit adder or subtractor has $3N$ nets with a fan-out of 2 each.

$$T_p = T_{CKO} + t_{net}^{2f,3N} + T_{ICK} = \left[2.5 + t_{net}^{2f,3N}\right] \text{ns} \tag{11}$$

4.2.2. Hybrid Adder/Subtractor. Hybrid adder/subtractor performs the addition or the subtraction between a redundant number and an unsigned one depending on the selection signal A/S. This circuit is composed by a PPM cell and a multiplexor to select between y_i or its one's complement (Fig. 4). Furthermore, it is needed to add a "1" in the least significant position (s_0^+) to complete the two's complement of Y when a subtraction is done. Each cell requires a CLB to be implemented (2 4-input functions), then, an N-bits adder/subtractor requires $N + 1/2$ CLBs.

The fixed delay in the critical path is the same as in the previous operators. Nevertheless, the routing delay is determined by the A/S net, which has a fan-out of $2N$. Obviously, the routing delay in this case will be longer than in the conventional adder/subtractor; it is given by:

$$T_p = T_{CKO} + t_{net}^{2Nf,1} + T_{ICK} = \left[2.5 + t_{net}^{2Nf,1}\right] \text{ns}. \tag{12}$$

4.2.3. Signed Digit Addition and Subtraction. The addition of two radix-2 redundant numbers (13) can be performed with a hybrid adder ($S_2' = X_2 + Y^+$) and a hybrid subtractor ($S_2 = S_2' - Y^-$). In the same way, the subtraction of two signed digit numbers (14) can be achieved by a hybrid subtractor ($S_2' = X_2 - Y^+$) and a hybrid adder ($S_2 = S_2' + Y^-$). Both operators are shown in Fig. 5.

$$S_2 = X_2 + Y_2 = X_2 + Y^+ - Y^-. \tag{13}$$

$$S_2 = X_2 - Y_2 = X_2 - Y^+ + Y^-. \tag{14}$$

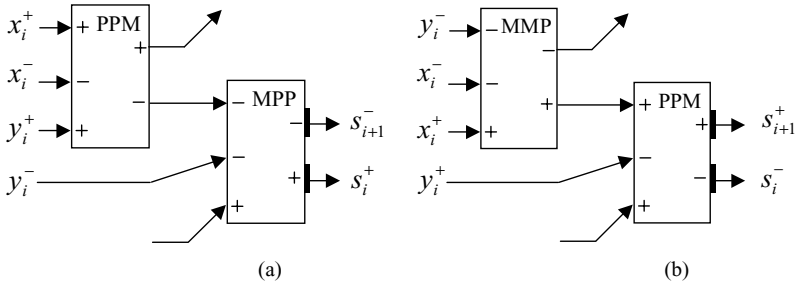

Figure 5. Signed digit addition (a) and subtraction (b).

There are two possibilities of dividing these cells in LUTs. The first one is as 2 4-input functions and the second is as 2 7-input functions (in the case of the adder the PPM cells are F functions and the MMP cells are H functions). Both possibilities needs 2 CLBs per cell, therefore, an N-bit signed digit adder or subtractor requires an area of $2N$ CLBs (4 times larger than the conventional ones).

The difference of these two implementations is found by analyzing the critical path. In the first method two routing delays are involved, one of the $3N$ nets with a fan-out of 2 determines the speed (15). In the second method only a routing delay is involved. The slowest delay is given by one of the $3N$ nets with a fan-out of 4 each (16). This last one achieves the highest frequency.

$$T_{\mathrm{p}} = T_{\mathrm{CKO}} + 2t_{\mathrm{net}}^{\mathrm{2f},3N} + T_{\mathrm{ILO}} + T_{\mathrm{ICK}} = \left[3.8 + 2t_{\mathrm{net}}^{\mathrm{2f},3N}\right] \mathrm{ns}. \quad (15)$$

$$T_{\mathrm{p}} = T_{\mathrm{CKO}} + t_{\mathrm{net}}^{\mathrm{4f},3N} + T_{\mathrm{IHCK}} = \left[3.3 + t_{\mathrm{net}}^{\mathrm{4f},3N}\right] \mathrm{ns}. \quad (16)$$

The previous operators can be easily pipelined by including FF in the outputs of the first hybrid module and in the input connected to the second module. By doing so each cell requires 4 registered 3-inputs LUTs and a FF, therefore the area increases by half a CLB. In this case the delay in the critical path is given in (17). As can it be seen the routing delay is determined by one of the $6N$ nets with a fan-out of 2, and is given by:

$$T_{\mathrm{p}} = T_{\mathrm{CKO}} + t_{\mathrm{net}}^{\mathrm{2f},6N} + T_{\mathrm{ICK}} = \left[2.5 + t_{\mathrm{net}}^{\mathrm{2f},6N}\right] \mathrm{ns}. \quad (17)$$

A 16-bit pipelined redundant adder or subtractor achieves a throughput of 110 MHz in a device with a speed degree of 1 with an area utilization lower than 5%.

Redundant arithmetic allows processing the data by MSDF manner. To perform a parallel MSDF redundant adder or subtractor it is necessary to latch the minus output of the PPM block and the y_i^- in the adder (Fig. 5(a)) or the plus output of the MMP cell and the y_i^+ in the subtractor (Fig. 5(b)). In that case, the operator requires 3 CLBs to be implemented (two 5-input functions, one 3-inputs function and one FF). For an N-bit operation the critical path is given by the slowest from $3N$ nets with a fan-out of 4 each:

$$T_{\mathrm{p}} = T_{\mathrm{CKO}} + t_{\mathrm{net}}^{\mathrm{4f},3N} + T_{\mathrm{IHCK}} = \left[3.3 + t_{\mathrm{net}}^{\mathrm{4f},3N}\right] \mathrm{ns}. \quad (18)$$

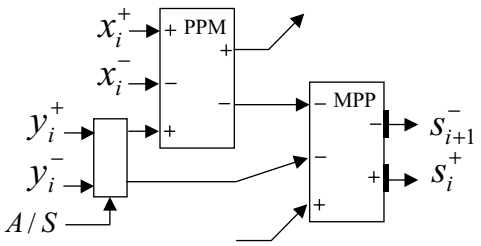

Figure 6. Signed adder/subtractor cell.

4.2.4. Signed Digit Adder/Subtractor. A signed digit adder/subtractor can be implemented with a signed digit adder by including logic to negate one of the digits when it is controlled by the signal A/S (Fig. 6). This negation can be performed in two ways. Either the positive and negative bit of the digit $-Y_2 = Y^- - Y^+$ have to be exchanged [28], as shown in Fig. 7(a), or by complementing to one both the positive and negative bits of each digit (Fig. 7(b)). From the FPGA implementation point of view the second option is better than the first one, since a switching box is decomposed by two 3-input functions while complementing the bits requires two 2-input functions. Therefore, this second option has been selected for the implementation.

Each cell of the adder/subtractor can be discomposed in four 4-input functions, as a consequence 2 CLBs are used. It is 4 times larger than the conventional one.

The critical path of an N-bit adder/subtractor contains two routing delays (19). The first one is given by the A/S net, which has a fan-out of $4N$ (it is present in each of the four functions).

$$T_{\mathrm{p}} = T_{\mathrm{CKO}} + t_{\mathrm{net}}^{4N\mathrm{f},1} + T_{\mathrm{ILO}} + t_{\mathrm{net}}^{\mathrm{2f}} + T_{\mathrm{ICK}} = \left[3.8 + t_{\mathrm{net}}^{4N\mathrm{f},1} + t_{\mathrm{net}}^{\mathrm{2f}}\right] \mathrm{ns}. \quad (19)$$

If this circuit is pipelined it can be divided into 2 4-input functions, 2 3-inputs functions and 1 2-input function. In this way each cell requires 2.5 CLBs while

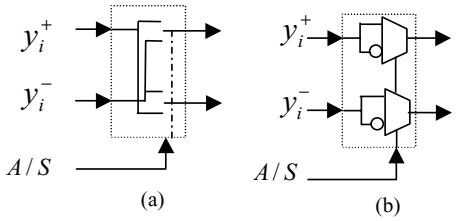

Figure 7. (a) Switching box; (b) complementing box.

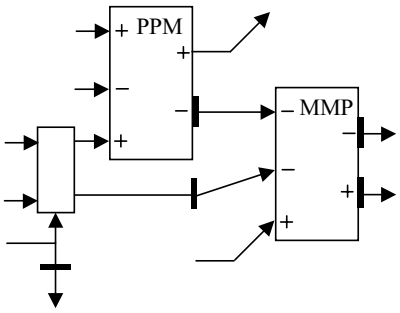

Figure 8. MSDF signed adder/subtractor.

enhancing the performance. A routing delay is avoided and less number of nets with a lower fan-out can produce the critical path (20).

$$T_{\mathrm{p}} = T_{\mathrm{CKO}} + t_{\mathrm{net}}^{3N\mathrm{f},1} + T_{\mathrm{ICK}} = \left[2.5 + t_{\mathrm{net}}^{3N\mathrm{f},1}\right] \text{ ns.} \quad (20)$$

The throughput achieved by a FPGA implemented 16-bit redundant adder/subtractor is 50 MHz. The A/S signal with a fan-out of 48 can be decomposed into two nets with lower fan-out by passing through an extra CLB.

The high fan-out of the A/S wire can be reduced if the circuit work with MSDF. In that case the A/S signal has to be latched before it is transmitted to the next cell (Fig. 8). The fan-out of A/S is now 3 (21). Nevertheless, its drawback is that the area of each cell increases to 4 CLBs.

$$\begin{aligned} T_{\mathrm{p}} &= T_{\mathrm{CKO}} + t_{\mathrm{net}}^{3\mathrm{f},1} + T_{\mathrm{ILO}} + t_{\mathrm{net}}^{2\mathrm{f},1} + T_{\mathrm{ICK}} \\ &= \left[3.8 + t_{\mathrm{net}}^{3\mathrm{f},1} + t_{\mathrm{net}}^{2\mathrm{f},1}\right] \text{ns} \end{aligned} \quad (21)$$

4.2.5. Most Significant Digit First Absolute Value. One of the CORDIC methods shown below needs the computation of the absolute value of a redundant number transmitted with MSDF.

The absolute value operation can be done in MSDF fashion if redundant numbers are used. The sign of a redundant number is given by the sign of the first non-zero digit. Then, beginning from the MSD, all digits are inspected until the first non-zero digit is found. The f_i signals are used to transmit the sign information from a digit to other. They can take the values ND, N or P (Non Decision, Negative or Positive) as shown in (22). In each digit its absolute value is achieved according to the information given by f_i and computed by (23). Initially $f_{-1} = \mathrm{ND}$ and at the end f_N takes the sign of the word.

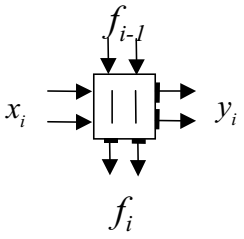

Figure 9. MSDF absolute value basic cell.

$$f_i = \begin{cases} f_{i-1} & \text{if } f_{i-1} = \mathrm{P} \text{ or } \mathrm{N} \\ \mathrm{P} & \text{if } f_{i-1} = \mathrm{ND} \text{ and } x_i = +1 \\ \mathrm{N} & \text{if } f_{i-1} = \mathrm{ND} \text{ and } x_i = -1 \\ \mathrm{ND} & \text{if } f_{i-1} = \mathrm{ND} \text{ and } x_i = 0 \end{cases} \quad (22)$$

$$y_i = \begin{cases} x_i & \text{if } f_i = \mathrm{P} \\ -x_i & \text{if } f_{i-1} = \mathrm{N} \\ x_i & \text{if } f_{i-1} = \mathrm{ND} \text{ and } x_i \geq 0 \\ -x_i & \text{if } f_{i-1} = \mathrm{ND} \text{ and } x_i = -1 \end{cases} \quad (23)$$

The basic cell that performs this task is shown in Fig. 9. Four 4-inputs functions are needed to implement each cell. Therefore, an N-bit MSDF absolute value operator requires $2N$ CLBs. The propagation delay is similar to (11) but it changes the routing delay. It is given by one of $4N$ nets with a fan-out of 4 each.

4.2.6. MSDF Absolute Value Hybrid Subtraction. The RM Differential CORDIC method (6.2) requires the computation of the absolute value of the hybrid subtraction working with MSDF. This operation is written in (24), where Z_2 and X_2 are redundant numbers and θ is a constant positive number.

$$Z_2 = |X_2 - \theta| \quad (24)$$

The basic cell required to perform this operation is shown in Fig. 10. There are two ways of mapping this

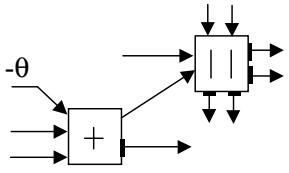

Figure 10. MSDF absolute value hybrid subtraction cell.

cell in Xilinx CLBs. The first one is performed by using four 5-inputs functions and one 2-inputs function, leading to 4.5 CLBs in each cell. The propagation time for an N-bit operation is shown in (25). The critical path is given by the slowest routing delay from $2N$ nets with a fan-out of 9 each.

$$\begin{aligned} T_{\mathrm{p}} &= T_{\mathrm{CKO}} + t_{\mathrm{net}}^{9\mathrm{f},2N} + T_{\mathrm{ICHK}} \\ &= \left[3.3 + t_{\mathrm{net}}^{9\mathrm{f},2N}\right] \mathrm{ns}. \end{aligned} \tag{25}$$

The second option is using four 4-inputs functions and two 2-inputs functions. In this way each cell requires 3 CLBs. Nevertheless, for an N-bit operation two routing delays are involved, although each one with lower fan-out, as indicated by:

$$\begin{aligned} T_{\mathrm{p}} &= T_{\mathrm{CKO}} + t_{\mathrm{net}}^{2\mathrm{f},2N} + T_{\mathrm{ILO}} + t_{\mathrm{net}}^{4\mathrm{f},1} + T_{\mathrm{ICK}} \\ &= \left[3.8 + t_{\mathrm{net}}^{2\mathrm{f},2N} + t_{\mathrm{net}}^{4\mathrm{f},1}\right] \mathrm{ns}. \end{aligned} \tag{26}$$

4.2.7. MSDF Absolute Value Redundant Subtraction. In VM Differential CORDIC (6.2) the absolute value of the subtraction of two redundant numbers processed in digit-parallel MSDF fashion is required (27).

$$Z_2 = |X_2 - Y_2| \tag{27}$$

The basic cell required to achieve this operation is shown in Fig. 11. There are two mapping possibilities. The first one uses four 4-inputs functions for the absolute value operator and two 5-inputs functions and a 3-inputs one for the subtractor. Therefore, the cell needs 4.5 CLBs to be implemented. The propagation

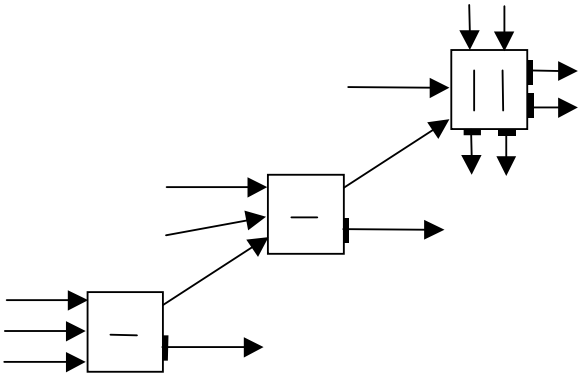

Figure 11. SMDF absolute value redundant subtraction cell.

delay for an N-bit operation contains two routings delays (28).

$$\begin{aligned} T_{\mathrm{p}} &= T_{\mathrm{CKO}} + t_{\mathrm{net}}^{2\mathrm{f},5N} + T_{\mathrm{IHO}} + t_{\mathrm{net}}^{4\mathrm{f}} + T_{\mathrm{ICK}} \\ &= \left[4.7 + t_{\mathrm{net}}^{2\mathrm{f},5N} + t_{\mathrm{net}}^{4\mathrm{f}}\right] \mathrm{ns}. \end{aligned} \tag{28}$$

The second possibility requires four CLBs (four 4-input functions and four 3-input functions). In this case three routing delays are involved in the propagation time (29).

$$\begin{aligned} T_{\mathrm{p}} &= T_{\mathrm{CKO}} + t_{\mathrm{net}}^{2\mathrm{f},3N} + T_{\mathrm{ILO}} + t_{\mathrm{net}}^{2\mathrm{f}} + T_{\mathrm{ILO}} + t_{\mathrm{net}}^{4\mathrm{f}} + T_{\mathrm{ICK}} \\ &= \left[5.1 + 2t_{\mathrm{net}}^{2\mathrm{f}} + t_{\mathrm{net}}^{4\mathrm{f}}\right] \mathrm{ns}. \end{aligned} \tag{29}$$

4.2.8. Converters. The conversion from conventional two's complement into redundant radix-2 arithmetic is a trivial operation which does not require any logic. All bits of the two's complement number except the sign bit are connected to x_i^+ inputs and the sign bit is connected to x_0^-. The others x_i^- and x_0^+ are zero.

The conversion from redundant into two's complement is an easy operation if all digits are being processed at a time. It consists in a conventional subtractor that computes $X_2 = X^+ - X^-$. Nevertheless, the realization of this operation decreases the performance because of its implicit carry propagation. The subtractor has to be pipelined to match its throughput with that in the rest of the circuit.

If the redundant digits are transmitted with the MSDF the conversion into two's complement is not a trivial operation. The problem consists in that it is not possible to know the value of any digit until the least significant one is available. The circuit to convert the digits in MSDF fashion is called On-the-Fly Converter [29], which can be seen in Fig. 12. It consists of two basic cells, the COPY and the APPEND cell [28]. The value of the current digit d_i is obtained by the sign of the next digit d_{i+1}. If d_{i+1} is zero the decision is postponed to iteration $i + 2$. A more detailed explanation of this circuit can be found in [28].

If this circuit is implemented on FPGA each APPEND cell requires four 2-inputs LUTs and each COPY cell requires two 4-inputs LUTs. The total area for an N-bit converter is given by $(N+5) \cdot N/2$ CLBs. The propagation time for a fully pipelined circuit involves two routing delays (30). The second one is due to the signal that are broadcast from an APPEND to all

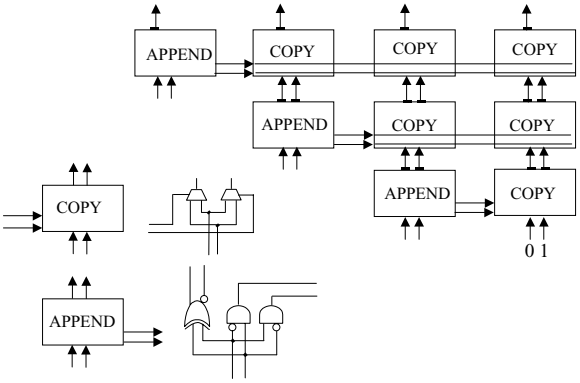

Figure 12. On-the-fly converter.

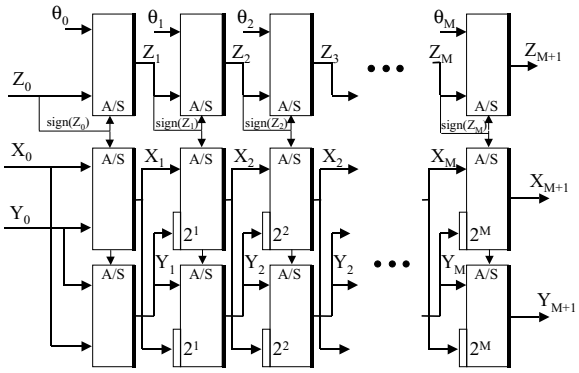

Figure 14. Pipelined bit-parallel CORDIC for rotation mode.

the COPY cells in the same row.

$$T_{\mathrm{p}} = T_{\mathrm{CKO}} + t_{\mathrm{net}}^{4\mathrm{f},2} + T_{\mathrm{ILO}} + t_{\mathrm{net}}^{(N-1)\mathrm{f},2} + T_{\mathrm{ICK}} = \left[3.8 + t_{\mathrm{net}}^{4\mathrm{f},2} + t_{\mathrm{net}}^{(N-1)\mathrm{f},2}\right] \mathrm{ns}. \quad (30)$$

A more efficient FPGA implementation of the previous circuit is obtained as shown in Fig. 13. By replicating the logic that selects the correct digit and by including it in each COPY cell the new cells called P and M are obtained. Each of these new cells requires one 4-input function. Therefore, the total area of the converter is given by $(N+3)\cdot N/2$ CLBs, i.e., N CLBs smaller than the previous one. Furthermore, in the propagation time of (31) one routing delay is avoided.

$$T_{\mathrm{p}} = T_{\mathrm{CKO}} + t_{\mathrm{net}}^{(N+1)\mathrm{f},2} + T_{\mathrm{ICK}} = \left[3.8 + t_{\mathrm{net}}^{(N+1)\mathrm{f},2}\right] \mathrm{ns} \quad (31)$$

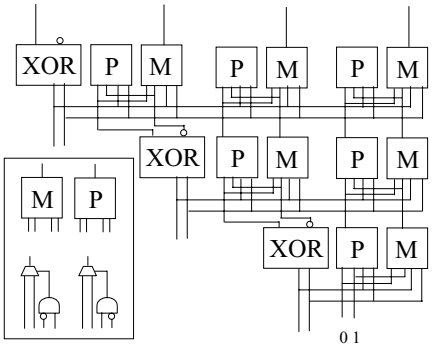

Figure 13. FPGA-based on-the-fly converter.

5. Conventional CORDIC on FPGA

The bit-parallel fully pipelined circuit to compute the equations in (1) for the Rotation Mode is shown in Fig. 14. It is based on ripple carry adders/subtractors (4.1.2). In each iteration the operation 2^{-i} is a hard-wired shift. The sign bit of Z_i is the line A/S, which indicates whether the next operation is an addition or a subtraction.

The whole processor requires $(2\cdot(M+1)\cdot(N+1)+(M+1)^2)/2$ CLBs. The critical path is fixed by one of the $M+1$ A/S nets. Each one of these nets has a fan-out of $2N+M+3$. The equation of the propagation time in the critical path is given by:

$$T_{\mathrm{p}} = T_{\mathrm{CKO}} + t_{\mathrm{net}}^{(2N+M+1)\mathrm{f},M+1} + T_{\mathrm{ASCY}} + (N-4)/2\cdot(T_{\mathrm{BYP}} + T_{\mathrm{NET}}) + T_{\mathrm{SUM}} = \left[6.5 + t_{\mathrm{net}}^{(2N+M+1)\mathrm{f},M+1} + (N-4)/2\cdot 0.45\right] \mathrm{ns}. \quad (32)$$

The latency of the processor is $M+1$ clock cycles.

In the case of a CORDIC implementation that computes either RM or VM with a precision of $M=16$ bits, the area occupied in the FPGA is 535.5 CLBs. The worst routing delay is given by one of the 17 nets with a fan-out of 61. The maximum operating frequency is about 25 MHz when it fits in a single chip with a speed degree of 1 and with an occupation less than 25%.

For a structure that computes both RM and VM the resources are 100% shared. In such a case it is only required to include a multiplexor to select the A/S signal between the sign bit of Z_i in RM or Y_i in VM. It only requires an extra of $(M+1)/2$ CLBs. Nevertheless, it slightly degrades the performance because an extra routing delay is added. The resulting critical path is

given in (33).

$$\begin{aligned} T_{\mathrm{p}} &= T_{\mathrm{CKO}} + t_{\mathrm{net}}^{1,17} + T_{\mathrm{ILO}} + t_{\mathrm{net}}^{(2N+M+1)\mathrm{f},M+1} \\ &\quad + T_{\mathrm{ASCY}} + (N-4)/2 \cdot (T_{\mathrm{BYP}} + T_{\mathrm{NET}}) + T_{\mathrm{SUM}} \\ &= \left[7.8 + t_{\mathrm{net}}^{1,17} + t_{\mathrm{net}}^{(2N+M+1)\mathrm{f},M+1} \right. \\ &\quad \left. + (N-4)/2 \cdot 0.45\right] \mathrm{ns}. \end{aligned} \tag{33}$$

6. Redundant CORDIC Methods

In order to solve the problems of the redundant arithmetic based CORDIC (commented in Section 2.1) several methods have been proposed. All of them maintain the scale factor constant. They have been classified in three groups: (i) those which are based on an estimation of the sign, (ii) the differential CORDIC algorithm and (iii) those based on the pre-computation of the directions of the micro-rotations.

6.1. Based on an Estimation of the Sign

Several methods perform an estimation of the sign by only exploring some digits of the word in each. These are the following:

- Double Rotation CORDIC [13].
- Correcting Rotation CORDIC ([13] for RM and [14] for VM).
- Householder [22–24].

Among these, the simplest implementation is the Correcting Rotation algorithm. The set of equations in this method is not the same for all iterations. If $D+1$ digits are inspected, in the iteration i, being $(i \bmod D) \neq 0$, the equations given in (34) are computed.

$$\begin{aligned} X_{i+1} &= X_i - \sigma_i \cdot 2^{-i} \cdot Y_i \\ Y_{i+1} &= Y_i + \sigma_i \cdot 2^{-i} \cdot X_i \\ Z_{i+1} &= Z_i - \sigma_i \cdot 2 \cdot \theta_i \end{aligned} \tag{34}$$

Nevertheless, if $(i \bmod D) = 0$ the set of equations to be computed is in (35)

$$\begin{aligned} X'_i &= X_i - \sigma_i \cdot 2^{-i} \cdot Y_i \\ Y'_i &= Y_i + \sigma_i \cdot 2^{-i} \cdot X_i \\ Z'_i &= Z_i - \sigma_i \cdot 2 \cdot \theta_i \\ X_{i+1} &= X'_i - \sigma_i \cdot 2^{-i} \cdot Y'_i \\ Y_{i+1} &= Y'_i + \sigma_i \cdot 2^{-i} \cdot X'_i \\ Z_{i+1} &= Z'_i - \sigma_i \cdot 2 \cdot \theta_i \end{aligned} \tag{35}$$

In the latter equations, $\sigma_i = -1$ for a negative sign otherwise $\sigma_i = 1$. To perform the operations for the X and Y data-path, the pipelined signed adder/subtractors are required (4.2.4). Although it is a little larger than the non-pipelined architecture it exhibits less routing delay. To compute the Z_i variable the hybrid adders/subtractors are needed (4.2.2), it also has to be pipelined in order to work synchronously with the X and Y operations. The 2^{-i} operations are also hardwired shifters.

The worst case is given when $D = 1$, only 2 digits are inspected. In that case all iterations compute the Eq. (35), hence, all iterations must be corrected. The occupied area corresponds to $5 \cdot N \cdot (M+1) + 2 \cdot (M+1)^2 + (M+1)/2$ CLBs. Since only two digits have to be inspected to estimate the sign, one 4-input function is required. The delay of the critical path is given in (36). It contains two routing delays and one of them consists of a very high fan-out $(6N + 2(M+1))$. Furthermore, there are $2(M+1)$ nets with that fan-out. The latency of the processor is $4(M+1)$ clock cycles.

In the case of a CORDIC with a precision of $M = 16$ bits ($N = 22$ bits) the area is 2448 CLBs (4.5 times larger than the conventional architecture) and there are 34 nets with a fan-out of 166. The latency is 64 clock cycles.

$$\begin{aligned} T_{\mathrm{p}} &= T_{\mathrm{CKO}} + t_{\mathrm{net}}^{1\mathrm{f},4} + T_{\mathrm{ILO}} \\ &\quad + t_{\mathrm{net}}^{(6N+2(M+1))\mathrm{f},2(M+1)} + T_{\mathrm{ICK}} \\ &= \left[3.8 + t_{\mathrm{net}}^{1\mathrm{f},4} + t_{\mathrm{net}}^{(6N+2(M+1))\mathrm{f},2(M+1)}\right] \mathrm{ns}. \end{aligned} \tag{36}$$

If only RM is required, the first routing delay in (36) can be avoided. It can be included in the pipelined hybrid adder/subtractor, and additionally it does not increase the required area. This is not true for the VM.

If a RM and VM CORDIC is desired the hardware is 100% shared. A multiplexor is required in each iteration to select the working mode; furthermore an extra routing delay and one T_{ILO} appear in (36).

Usually this method is used by inspecting at least 3 digits ($D = 2$). Although it reduces area in respect to the previous structure the throughout is deteriorated, an extra routing delay and one T_{IHO} appear in (36). It is caused by the sign estimation operation (a 6-input function), which is accomplished by 2.5 CLBs.

The other methods enumerated in this section have the same problems as the Correcting Rotation CORDIC. Furthermore, they have more complicated equations which lead to more area and less throughput.

6.2. *Differential CORDIC*

Differential CORDIC (DCORDIC) [17] is a redundant arithmetic based method that keeps a constant scale factor without including additional rotations. In DCORDIC, the directions of the micro-rotation are chosen such that the remaining rotation angles in RM (37) and Y_i variable in VM (38) always decreases. The goal of this method is the MSDF computation of all involved operations. The sign in one iteration is not necessary to begin to process the next operation.

$$|Z_{i+1}| = \|Z_i| - \theta_i| \tag{37}$$

$$|Y_{i+1}| = \|Y_i| - 2^{-i} \cdot X_i| \tag{38}$$

In RM DCORDIC the iteration variable is called $\hat{Z}_i$ with $|\hat{Z}_i| = |Z_i|$. The directions of the micro-rotations are obtained by computing (39).

$$\begin{aligned} |\hat{Z}_{i+1}| &= \|\hat{Z}_i| - \theta_i| \\ \sigma_i = \text{sign}(Z_{i+1}) &= \text{sign}(Z_i) \cdot \text{sign}(\hat{Z}_{i+1}) \end{aligned} \tag{39}$$

Then, one N-bit MSDF absolute value hybrid subtractor (4.2.6) is required to compute the Z_i variable in each iteration. Such structure also needs a skew delay at the input to arrive the data in MSDF fashion. To obtain each σ_i, an XOR gate is utilized.

Similar to the Correcting method, the X_i and Y_i are computed (1), (34). This can be achieved by using the pipelined signed adder/subtractor or Carry Save ones. In this case, these variables are not processed with MSDF, which allows the 2^{-i} operation to be performed with hard-wired shifters. Two delay matrixes are needed to synchronize the X and Y computation with the σ_i's.

The occupied area by an M-bits precision processor is:

- Skew matrix: $M \cdot (M-1)/4$ CLBs
- MSDF absolute value hybrid subtractor data-path: $3 \cdot (M+1)^2$ CLBs
- Computation of σ_i: $3 \cdot M + 1$ CLBs
- Delay Matrixes: $N \cdot (M+1)$ CLBs
- Computation of X and Y variables: $5 \cdot N \cdot (M+1)$ CLBs

The latency achieved will be $3 \cdot (M+1) + 1$ clock cycles. The propagation delay in the computation of Z_i is given by (26); however, in this case there are $2 \cdot N \cdot (M+1)$ nets with that equation. The critical path is determined by one of the $M+1$ nets which drives the A/S signal in the X and Y data-path. Each of those nets has a fan-out of $6N$ (40).

$$T_{\text{p}} = T_{\text{CKO}} + t_{\text{net}}^{6N\text{f},M+1} + T_{\text{ICK}} \tag{40}$$

In case of a RM DCORDIC with $M = 16$ bits precision the occupied area is 2883 CLBs. Its latency corresponds to 52 clock cycles. It achieves a throughput of 11 MHz in a device with speed degree of 1 and a 91% of area utilization. This is due to one of the 17 nets with a fan-out of 132.

The high fan-out that causes the throughput decrement can be avoided if the X and Y are computed with MSDF. Nevertheless if this is realized the operator 2^{-i} can no longer be implemented using hard-wired shifters, but with barrel shifters which require larger area.

In VM DCORDIC the iteration variable is named $\hat{Y}_i$ with $|\hat{Y}_i| = |Y_i|$. The direction of the micro-rotations is achieved by computing Y_i as indicated in (41).

$$\begin{aligned} |\hat{Y}_{i+1}| &= \|\hat{Y}_i| - 2^{-i} \cdot \hat{X}_i| \\ \hat{X}_{i+1} &= \hat{X}_i + 2^{-i} \cdot |\hat{Y}_i| \\ \sigma_i = \text{sign}(Y_{i+1}) &= \text{sign}(Y_i) \cdot \text{sign}(\hat{Y}_{i+1}) \end{aligned} \tag{41}$$

To compute the Y_i variable an MSDF absolute value redundant subtractor (4.2.7) is required. The X_i variable is processed by using redundant adders in MSDF. These adders require an extra level of pipelining in order to have the same latency as the circuit that computes the Y_i variable. It is also necessary to include one skew delay in each one of these two data-paths to allow the computation with MSDF manner. The resulting X data has to be converted into two's complement representation by an On-the-Fly Converter (4.2.8). One significant difference compared to all the previous structures is that the 2^{-i} operations can not be done by hard-wired shifters but with larger area consuming barrel shifter.

The computation of the angle is performed according to (1), and by processing all digits at a time. Therefore, in each iteration a hybrid adder/subtractor (4.2.2) is required. Furthermore, each operator needs two extras stages of pipelining to be synchronized with the decision path.

The area occupied by this processor corresponds to:

- Skew delays: $M \cdot (M-1)/2$ CLBs
- Y data-path: $4 \cdot N \cdot (M+1)$ CLBs
- X data-path: $4 \cdot N \cdot (M+1)$ CLBs

- Barrel shifters: $(M \cdot (M+1) \cdot (3 \cdot N - 2 \cdot M - 1))/6$ CLBs
- Z data-path: $3 \cdot N \cdot (M+1)$ CLBs
- On-the-Fly Converter: $(M+3) \cdot M/2$ CLBs

This processor has a latency of $3 \cdot (M+1) + N$ clock cycles. The critical path in the decision circuit is given by one of the $3 \cdot N \cdot (M+1)$ nets with a propagation delay as indicated in (29). In the Z_i computation path it is determined by the slowest of $M+1$ nets with a fan-out of $2 \cdot (M+1)$, as written in (42).

$$T_{\mathrm{p}} = T_{\mathrm{CKO}} + t_{\mathrm{net}}^{2(M+1)\mathrm{f},M+1} + T_{\mathrm{ICK}} \tag{42}$$

For a precision of $M = 16$ bits, the VM DCORDIC requires 5803 CLBs. Its latency is 73 clock cycles. The achieved throughput is 18 MHz when it is implemented in a chip with speed degree of 0.9 and with an occupation degree of 82%. The critical path is given by one of the 17 nets with a fan-out of 34.

In a shared RM and VM DCORDIC structure the resources can not be shared. The only way to share resources in this structure is by not using MSDF processing. However, in that case a carry propagation is required to obtain the sign. Therefore, there is no advantage in its utilization.

6.3. Pre-Computed CORDIC (P-CORDIC)

Another method to perform CORDIC algorithm is by directly computing the direction of the micro-rotation without iterations. In [25, 26] the correlation between the direction of the micro-rotation and the input angle is obtained for the RM. If the directions are coded in Offset Binary Codec (OBC):

$$d = a_0 \cdot a_1 a_2 \cdots a_{N-1} \tag{43}$$

with $a_i \in \{-1, 1\}$. If a_i takes the value 1 the iteration i performs a positive micro-rotation and if a_i is -1, a negative one is performed.

The correlation between the input angle Z and the direction of all micro-rotation is given by (44).

$$d = 0.5 \cdot Z + \mathrm{sign}(Z) \cdot b + \mathrm{sign}(Z) \cdot \varepsilon_i \tag{44}$$

Consequently, a RM CORDIC processor is composed by a block that computes (44) and another block, which executes the micro-rotations of the X and Y variables. This last one is performed in the same way that in the previous methods, that is with a redundant adder/subtractor or a Carry Save one. Therefore, that block has the same occupation than it has in the RM DCORDIC and its critical path is given by (40).

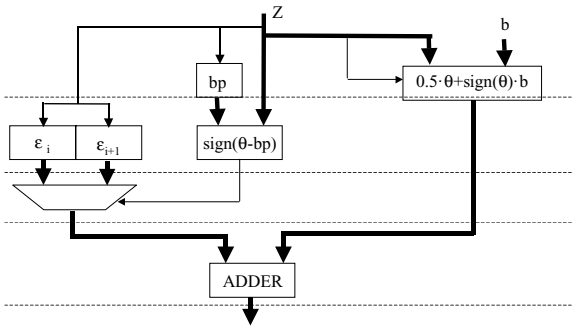

Figure 15. Block diagram of the correlation computation.

The computation of the micro-rotations directions for an FPGA implementation is presented as a block diagram in Fig. 15. According to the input Z a break point (bp) is chosen and subtracted from Z. Depending on its sign either the discontinuity either ε_i or ε_{i-1} is selected and added to the result of $Z + \mathrm{b}$, where b is a constant.

For a precision of M-bit the occupation of this block is $6.5 \cdot M$ CLBs. Furthermore, a skew delay is needed to distribute the directions of the micro-rotations into the X and Y computation block. Therefore, other $(M+1) \cdot M/4$ CLBs are included. In the case of a high speed FPGA implementation having all the direction of the micro-rotations at a time has no advantage.

If only the area of the decision block is considered it is 1.6 times larger than the conventional CORDIC implementation. This method can be faster and less area consuming than the other redundant ones but slower and larger when compared with the conventional arithmetic.

If area is the most important issue this block can be implemented without pipelining. In such a case its occupation is $5 \cdot M$ CLBs (1.7 times lower than the conventional case). Therefore, the X and Y computation could be performed with conventional arithmetic based on a non-pipelined structure which requires $(M+1) \cdot N$ CLBs.

The correlation between the directions and the X and Y input in VM is obtained in [25]. This method has been ruled out for FPGA design because its implementation involves two MSDF multiplications and one MSDF addition. But as it was already seen the implementation of MSDF components require more area.

Table 2. FPGA implementation of CORDIC algorithms.

Algorithm	Number of CLBs	Occupation degree (%)	Fan-out in critical path	Number of nets	Routing delay (ns)	Propagation delay (ns)	Maximum frequency (MHz)	Latency (clock cycles)	(μs)
CORDIC (RM , VM)	535.5	22	61	17	24.4	40	25	17	0.68
RM Double Rotation ($D = 1$)	2448	77.2	166	34	96,2	100	10	68	6.8
DCORDIC (RM)	2883	91	132	17	86.2	90.9	11	52	4.7
DCORDIC (VM)	5803	82	34	17	54.2	55.5	18	73	4.05

7. Discussion of the FPGA Implementisation

Conventional arithmetic-based CORDIC, double rotation and DCORDIC algorithm, have been modeled on VHDL for a precision of 16 bits and have been implemented on Xilinx XC4000 FPGAs. The same die area has been used to implement Conventional CORDIC, Double Rotation CORDIC and DCORDIC RM, however, DCORDIC VM does not fit in the device used in previous designs. The results of the implementation are summarized in Table 2. In this table the number of used CLBs, the occupation degree of the chip, the fan-out of the critical path to determine its routing delay and the number of nets that have the maximum number of fan-outs are shown. The routing delay and the total propagation delay of the critical path is obtained from the simulation after placement and routing process and are also mentioned in the table. Taking into account the number of clock cycles and the propagation delay of the critical path, the maximum frequency and the latency of the different CORDIC implementation can be obtained.

As it can be seen in Table 2, the routing delay always dominates the total propagation delay and follows closely the relationship established in Section 3, in which was indicated that nets with a higher fan-out have a longer propagation delay. Nevertheless, the implementation of the DCORDIC in vectoring mode shows the smallest number of fan-outs in the critical path (almost 50% of the conventional implementation) while the propagation delay is more than twice as large. This is due to the four times higher degree of occupation which makes if difficult for XILINX to route all the interconnects of the chip. Our simulations support the conclusion that the conventional CORDIC architecture achieves the shortest propagation delay and hence the fastest implementation. The CORDIC architectures based on redundant arithmetic require at least five times the amount of CLB resulting in a denser FPGA which makes the routing more difficult. The simulation results also support the conclusion that for implementations with the same degree of occupation the FPGA with the larger number of nets of a higher fan-out results in a longer propagation delay (see row 2–4 in Table 2).

Generally, the routing delay cannot be determined until an architecture is placed and routed. Simulation can be performed to obtain the critical path of the design. The propagation delay of the critical path is highly dependent on the placement of the cells. The process of synthesis, placement and routing normally takes quite a long time. Hence, an easier decision criteria can be used by taking the number of fan-outs, the number of nets with this fan-out and the numbers of CLBs into consideration. With this information, a quicker decision can be made which implementation will result in the most efficient FPGA implementation.

8. Performance Enhancement of FPGA-Based CORDIC

From the data in previous sections it can be said that the fastest FPGA-based CORDIC can be achieved by using conventional arithmetic. All the FPGA-based redundant arithmetic CORDIC processors require at least 5 times larger area and are slower because of mainly the very high fan-out of the A/S signals. Therefore, to enhance the throughput of FPGA-based CORDIC more efforts need to directed towards improving the performance of the conventional arithmetic structure.

To enhance performance in conventional CORDIC several strategies can be considered: to reduce the fan-out of A/S wires, to reduce the carry propagation of the adder/subtractor by mean of pipelining and optimizing for the application.

8.1. Fan-Out Reduction

One way to reduce the fan-out in conventional CORDIC implementation is to increase the length of

the adder/subtractors, that process Z in RM or Y in VM, by two bits. Then, the extra two output-bits will contain an extension of the sign bit. Each one of these three sign bits can be used to drive a different adder/subtractor of the previous stage. Therefore, the fan-out of each A/S net is divided by three with respect to the case in Section 5.

The area costs of this method in an M-bit precision CORDIC is $3 \cdot (M + 1)/2$ extra CLBs. A RM conventional CORDIC with 16-bits of precision implemented in a FPGA with speed degree of 1 can achieve a throughput of 29 MHz with an occupation of 552 CLBs.

8.2. Pipelining

In each adder/subtractor the carry propagation can be decreased by pipelining. In this case the pipelining does not only reduce the carry propagation but also the fan-out of the A/S signals (twice A/S nets have half the fan-out). However, the drawback of the pipelining stages is an increase in area by a factor of 2.5.

A pipelined conventional RM CORDIC implemented in an FPGA with speed degree of 1 with a 50% of area utilization achieves a frequency of 55 MHz with an area of 1340 CLBs.

8.3. Tailoring for the Application

The reconfiguration feature of FPGAs allows adapting the processor structure to the requirement of the target application.

If only RM or VM is required a method to reduce the area is achieved by taking profit of some CORDIC algorithm characteristics. For example it is known that all variables converge to specific values and hence, in each iteration the word-length of the data-paths can be reduced (triangle shape) [30, 31].

Another example is found if a fixed angle RM CORDIC is required. The algorithm can be directly mapped without using adder/subtractor but using an adder or a subtractor in each stage. In such a case not only the area is reduced but the throughput is also increased (the high fan-out of the adder/subtractors is avoided).

9. Conclusions

In this paper FPGA implementations of the fastest full custom methods for the CORDIC algorithm have been evaluated. Since all these methods are based on redundant arithmetic, the FPGA implementation of the required operators to perform the different CORDIC methods has been studied. It can be concluded that the redundant arithmetic operators have a 4 to 5 times larger occupation than the conventional architecture. Furthermore, in FPGA implementation the speed advantages of the full custom design has been lost due to the longer routing delays caused by the increase of the fan-out and the number of nets. Therefore, the redundant arithmetic based CORDIC methods are not suitable for FPGA implementation. All these methods have at least a 4 times larger occupation and obtain a lower throughput compared to the conventional one. As a consequence, the fastest and most area-efficient FPGA implementation of CORDIC is achieved by using the conventional arithmetic. Additionally, if a higher throughput is required, it can be achieved either by pipelining the previous structure or limiting the fan-out by extending the sign bit to use different wires to drive the A/S signal in each adder/subtractor. These conclusions have been proof for circular CORDIC implementations, however, they can be extended to linear and hyperbolic CORDICs.

References

1. P.J. Graumann and L.E. Turner, "Implementing Digital Signal Processing Algorithms Using Pipelined Bit-Serial Arithmetic and Field Programmable Gate Arrays," in *First International ACM/SIGDA Workshop on Field Programmable Gate Arrays (FPGA'92)*, 1992.
2. J. Isoaho, J. Pasanen, O. Vainio, and H. Tenhunen, "DSP System Integration and Prototyping with FPGAs," *Journal of VLSI Signal Processing*, vol. 6, 1993, pp. 155–172.
3. M. Wahab and D. Puckey, "FPGA-Based DSP Systems," in *Abindon EE&CS books*, W.R. Moore and W. Luk (Eds.), 1994.
4. R.J. Petersen and B. Hutchings, "An Assessment of the Suitability of FPGA-Based Systems for Use in DSPs," in *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1995, no. 975, pp. 293–302.
5. J.E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Trans. Electronic Computers*, vol. EC-8, no. 3, 1959, pp. 330–334.
6. J.S. Walther, "A Unified Algorithm for Elementary Functions," in *Proc. Spring. Joint Comput. Conference*, vol. 38, 1971, pp. 379–385.
7. Y. Hu, "CORDIC-Based VLSI Architectures for Digital Signal Processing," *IEEE Signal Processing Magazine*, vol. 9, no. 3, July 1992, pp. 16–35.
8. R. Andraka, "A Survey of CORDIC Algorithms for FPGAs," in *Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays (FPGA '98)*, Monterey, CA, Feb. 22–24, 1998, pp. 191–200.
9. U. Meyer-Base, A. Meyer-Base, and W. Hilberg, "COordinate Rotation DIgital Computer (CORDIC) Synthesis for FPGA," in

4*th International Workshop on Field Programmable Logic and Applications (FPL'94*), Prag, Czech Republic, 7–9 September, 1994.
10. C. Dick, "Computing the Discrete Fourier Transform on FPGA Based Systolic Arrays," in *ACM/SIGDA Int. Symp. on Field Programmable Gate Array*, Feb. 1996, pp. 129–135.
11. U. Meyer-Base, A. Meyer-Base, J. Mellott, and F. Taylor, "A Fast Modified CORDIC- Implementation of Radial Basis Neural Networks," *Journal of VLSI Signal Processing*, vol. 20, 1998, pp. 211–218.
12. M.A. Mayosky, P.E. Battaiotto, and G.M. Toccaceli, "A CORDIC Architecture for Vector Control," in *Proc. of the Int. Conf. on Signal Processing Applications and Technology*, 1998.
13. N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC Methods with a Constant Scale Factor for Sine and Cosine Computation," *IEEE Transactions on Computers*, vol. 40, no. 9, 1991.
14. J.-A. Lee and T. Lang, "Constant-Factor Redundant CORDIC for Angle Calculation and Rotation," *IEEE Transactions on Computers*, vol. 41, no. 8, 1992.
15. H. Lin and A. Sips, "On-Line CORDIC Algorithms," *IEEE Transactions on Computers*, vol. 39, 1990, pp. 1038–1052.
16. J. Duprat and J.-M. Muller, "The CORDIC Algorithm: New Results for Fast VLSI Implementation," *IEEE Transactions on Computers*, vol. 42, 1993, pp. 168–178.
17. H. Dawid and H. Meyr, "The Differential CORDIC Algorithm: Constant Scale Factor Redundant Implementation without Correcting Iterations," *IEEE Transactions on Computers*, vol. 45, no. 3, 1996.
18. S. Wang, V. Piuri, and E. Swartzlander, "Hybrid CORDIC Algorithms," *IEEE Transactions on Computers*, vol. 46, 1997, pp. 1202–1207.
19. C. Li and S. Chen, "A Radix-4 Redundant CORDIC Algorithm with Fast On-Line Variable Scale Factor Compensation," in *Int. Symposium of Circuit and Systems (ISCAS'97)*, Hong Kong, Jun. 1997, pp. 639–642.
20. R.R. Osorio, E. Antelo, J. Bruguera, and E. Zapata, "Digit On-Line Large Radix CORDIC Rotator," in *Int. Conf. On Application-Specific Array Processors*, Strasbourg, France, Jul. 1995, pp. 247–257.
21. J. Villalba, J. Hidalgo, E. Zapata, E. Antelo, and J. Bruguera, "CORDIC Architectures with Parallel Compensation of the Scale Factor," in *Int. Conf. on Application-Specific Array Processors*, Strabourg, France, Jul. 1995, pp. 258–269.
22. Shen-Fu Hsiao and Jean-Marc Delosme, "Householder CORDIC Algorithm," *IEEE Transactions on Computers*, vol. 44, no. 8, 1995.
23. Shen-Fu Hsiao and Jen-Yin Chen, "Design, Implementation and Analysis of a New Redundant CORDIC Processor with Constant Scaling Factor and Regular Structure," *Journal of VLSI Signal Processing*, vo. 20, 1998, pp. 267–278.
24. Shen-Fu, Hsiao, "A High-Speed Constant-Factor Redundant CORDIC Processor without Extra Correcting or Scaling Iterations," in *IEEE Int. Conf. On Circuits and Systems (ISCAS'99)*, Florida, 1999.
25. M. Kuhlmann and K.K. Parhi, "A High-Speed CORDIC Algorithm and Architecture for DPS Applications," in *Proc. of the 1999 IEEE Workshop on Signal Processing Systems (SiPS'99)*, Taipei, Taiwan, Oct. 1999.
26. M. Kuhlmann and K.K. Parhi, "A New CORDIC Rotation Method for Generalized Coodinate Systems," in *Proc. of the 1999 Asilomar Conference on Signal, Systems and Computers*, Pacific Grove, CA, Oct. 1999.
27. J. Moran, I. Rios, and J. Meneses, "Signed Digit Arithmetic on FPGAs," in More FPGAs, pp. 251–261, W.R. Moore and W. Luk (Eds.), Abindon EE&CS books, 1994.
28. K.K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, Wiley and Sons, 1999.
29. M.D. Ercegovac and T. Lang, "On-the-Fly Conversion of Redundant into Conventional Representations," *IEEE Transactions on Computers*, vol. 36, 1987, pp. 895–897.
30. D. Timmermann and S. Dolling, "Unfolded Redundant CORDIC VLSI Architectures with Reduced Area and Power Consumption," in *Int. Conf. on VLSI 97*, Brasilia, Sept. 1997.
31. A. Wassatsch, S. Dolling, and D. Timmermann, "Area Minimization of Redundant CORDIC Pipeline Architectures," in *IEEE International Conference on Computer Design (ICCD'98)*, Austin, Texas, October 1998.

Javier Valls received the Telecommunication Engineering degree from the Universidad Politecnica de Cataluña, and the Ph.D. degree in Telecommunication Engineering from the Universidad Politecnica de Valencia, Spain in 1993 and 1999, respectively. He is Assistant Professor at the Department of Electronics from Universidad Politecnica de Valencia since 1996. Current research interests include the design of FPGA-based systems, computer arithmetic, VLSI signal processing and digital communications.
jvalls@eln.upv.es

Martin Kuhlmann received his Diplom Ingenieur and Ph.D. degree in Electrical Engineering from the University of Technology Aachen, Germany in 1997 and from the University of Minnesota in 1999, respectively. Currently, he is a staff design engineer at Broadcom Corporation, Irvine, CA. His research interests include computer arithmetic, digital communication, VLSI design and deep-submicron crosstalk. He is a member of IEEE.
kuhlmann@broadcom.com

Keshab K. Parhi is a distinguished McKnight University Professor of Electrical and Computer Engineering at the University of Minnesota, Minneapolis where he also holds the Edgar F. Johnson Professorship. He is currently a senior principal scientist at Broadcom Corp., Irvine, on leave of absence. He received the B.Tech., M.S.E.E., and Ph.D. degrees from the Indian Institute of Technology, Kharagpur (India) (1982), the University of Pennsylvania, Philadelphia (1984), and the University of California at Berkeley (1988), respectively.

His research interests include all aspects of VLSI implementations of broadband access networks. He is currently working on VLSI adaptive digital filters, equalizers and beamformers, error control coders and cryptography architectures, low-power digital systems, and computer arithmetic. He has published over 320 papers in these areas. He has authored the text book *VLSI Digital Signal Processing Systems* (Wiley, 1999) and co-edited the reference book *Digital Signal Processing for Multimedia Systems* (Dekker, 1999).

He received a Golden Jubilee medal from the IEEE Circuits and Systems Society in 1999, the 2001 W.R.G. Baker Prize Paper award from the IEEE, a 1996 Design Automation Conference best paper award, the 1994 Darlington and the 1993 Guillemin-Cauer best paper awards from the IEEE Circuits and Systems society, the 1991 paper award from the IEEE signal processing society, the 1991 Browder Thompson prize paper award from the IEEE, and the 1992 Young Investigator Award of the National Science Foundation.

Dr. Parhi has served on editorial boards of the IEEE Transactions on Circuits and Systems, the IEEE Transactions on Signal Processing, the IEEE Transactions on Circuits and Systems—PART II: Analog and Digital Signal Processing, the IEEE Transactions on VLSI Systems, and the IEEE Signal Processing Letters. He is an editor of the Journal of VLSI Signal Processing. He is the guest editor of a special issue of the IEEE Transactions on Signal Processing, two special issues of the Journal of VLSI Signal Processing and a special issue of Journal of Analog Integrated Circuits and Signal Processing. He served as technical program co-chair of the 1995 IEEE Workshop on Signal Processing and the 1996 ASAP conference. He serves on numerous technical program committees, was a distinguished lecturer of the IEEE Circuits and Systems Society (1994–1999), and is a fellow of IEEE.

parhi@broadcom.com