


Article

Design of Cascaded CORDIC Based on Precise Analysis of Critical Path

Pramod Kumar Meher¹ and Sang Yoon Park^{2,*} ¹ C. V. Raman College of Engineering, Bhubaneswar 752 054, India; pkmeher@gmail.com² Department of Electronic Engineering, Myongji University, Yongin 17058, Korea

* Correspondence: syark@mju.ac.kr; Tel.: +82-31-330-6751

Received: 22 February 2019; Accepted: 26 March 2019; Published: 29 March 2019



Abstract: A conventional coordinate rotation digital computer (CORDIC) has a low throughput rate due to its recursive implementation of micro-rotations. On the contrary, a fully-pipelined cascaded CORDIC provides a very high throughput rate at the cost of high complexity and large area. In this paper, possible design choices of cascaded CORDIC are explored over a wide range of operating frequencies, throughput rates, latency, and area complexity. For this purpose, we present a fine-grained critical path analysis of the cascaded CORDIC in terms of bit-level delay. Based on the propagation delay estimate, we propose an algorithm for determining the required number of pipeline stages and locations of the pipeline registers in order to meet the time constraint in a particular application. A hybrid cascaded-recursive CORDIC is also proposed to increase the throughput rate, and to reduce the latency and energy per sample (EPS). From synthesis results, we show that the proposed pipelined cascaded CORDIC with only four pipeline stages requires 31.1% less area and 29.0% less EPS compared to a fully-pipelined CORDIC. An eight stage pipelined recursive cascaded CORDIC provides 18.3% less EPS and 40.4% less area-delay product than a conventional CORDIC.

Keywords: coordinate rotation digital computer; pipelining; parallel; digital arithmetic; digital signal processing; VLSI

1. Introduction

The key concept of coordinate rotation digital computer (CORDIC) arithmetic is that it can effectively compute trigonometric functions, vector rotation, multiplication, and division through an iterative formulation of shift and add operations. Ever since the CORDIC algorithm was first described in 1959 by Jack E. Volder [1,2], a wide variety of explorations of CORDIC applications, algorithms, and architectures have been investigated to find low-cost, high-performance hardware solutions [3–12].

Throughput and power consumption have become the major issues in CORDIC algorithm design, especially for implementations in embedded systems with limited resources. As CORDIC acts as a basic arithmetic operator in addition to adders, subtractors, and multipliers in hardware implementation, their performance greatly affects the performance of the entire system, especially in CORDIC applications requiring high performance, such as fast Fourier transform (FFT) [13] or embedded FPGA-based synthesizers, including chaotic Pseudo-random number generators [14]. In particular, the latency of CORDIC computation is a major issue due to the large number of iterations required to preserve sufficient precision of the output, despite its linear-rate convergence [15]. Therefore, the speed of CORDIC operations is limited either by the required precision (number of iterations) or the clock period. Angle recoding (AR) methods [5,6,8–10] can be used to reduce the number of CORDIC iterations by encoding the rotation angle as a linear combination of a set of selected elementary angles. However, selective implementation of micro-rotation carries significant scaling overhead.

Parallel processing and pipelining are two popular techniques used to enhance the performance of any computing system [16]. Area is traded for speed in parallel processing while latency and register complexity are traded for speed in pipeline processing [17,18]. The scope of parallel processing and pipelining for improving the performance of CORDIC has been explored in the literature [9,19–21]. Due to the inherent sequential nature of CORDIC, it is not possible to have a full parallel CORDIC, and any parallel implementation of CORDIC carries significant overhead. In [9], it was further shown that the rotation direction can be determined once the input angle is known in order to enable parallel realization of micro-rotations. Despite the parallel execution of CORDIC rotation according to [9], the decomposition method of each positional binary weight produces extra micro-rotation stages, especially as the word length of the input angle increases.

An alternative approach to speed up CORDIC is to reduce the clock period and increase the throughput rate of the CORDIC output. As the CORDIC iterations are identical and involve nearly the same complexity, mapping them onto pipelined architectures is very hardware-friendly. The main objective of pipelined implementation is to reduce the critical path. The pipelined architecture of CORDIC was suggested by Deprettere et al. in 1984 [21]. Pipelined CORDIC circuits were used thereafter for high-throughput sinusoidal wave generation, FFT, adaptive filters, and other signal processing applications [22–24]. Given each pipeline stage performs one predetermined micro-rotation, the number of shifts required for shift-add or shift-sub operations in each pipeline stage is known a priori. Therefore, shift operations can be hardwired, and barrel-shifters can be completely eliminated in the non-recursive cascaded implementation of CORDIC. Then, the critical path of pipelined CORDIC amounts to the time required to execute add or subtract operations in each stage.

Fully-pipelined-cascaded CORDIC (FPCC) has potential for a very low clock period and very high throughput. However, such high throughput rate is usually not required in real applications. For example, when CORDIC is used to implement the complex butterfly operation for a fast Fourier transform (FFT), the throughput rate provided by CORDIC must match the throughput requirement of the application and the type of FFT implementation, such as fully-pipelined implementation, folded-pipeline implementation, and the number of butterfly circuits. On the other hand, we find that removing a pair of pipeline registers to merge two pipeline stages does not substantially increase the critical path. The propagation delay of a pair of CORDIC stages is generally assumed to be the sum of delays of two adders or two subtractors. However, such an assumption is currently invalid in ASIC and FPGA implementation, wherein adders or subtractors in each stage are not considered as discrete components. We also find that the rear stages in the cascaded CORDIC have longer propagation delays than the preceding stages, thus inserting pipeline registers for each is not a good pipelining strategy for a given number of stages. However, there is no systematic analysis regarding the appropriate pipeline decision in a cascaded CORDIC. In this paper, we discuss the fine-grained estimation of propagation delays in cascaded CORDIC. Based on that, we derive a formulation that could be used by a designer to determine the number of pipeline stages to be considered and where to place the pipeline registers in the cascaded design.

Cascaded CORDIC involves one CORDIC unit for each micro-rotation, while at the other extreme, recursive CORDIC uses only one CORDIC unit for all micro-rotations. Instead, we can have a hybrid of non-recursive cascaded CORDIC and recursive CORDIC, which we refer to as pipelined-recursive-cascaded CORDIC (PRCC). PRCC consist of a cascade of a few CORDIC units, where each such CORDIC unit performs a certain number of micro-rotations recursively. In this paper, we investigate the design and implementation of such recursive cascade as another design option for the cascaded CORDIC.

The rest of the paper is organized as follows. A critical path analysis for a cascaded CORDIC with unknown rotation angles is presented in Section 2. The fine-grained critical path analysis of a cascaded CORDIC for known rotation angles is presented in Section 3. An algorithm for determining the number of pipeline registers and their locations for a given timing constraint is also proposed in Section 3. A hybrid recursive cascaded CORDIC is presented in Section 4. The performance of

the proposed designs in terms of area, throughput, latency, and power consumption is discussed in Section 5. Conclusions are given in Section 6.

2. Critical Path Analysis of Cascaded CORDIC for Unknown Rotation Angles

The rotation mode of the CORDIC algorithm performs vector rotation iteratively, as follows [1,2]:

$$x_{n+1} = x_n - \text{sign}(\omega_n) \cdot y_n \cdot 2^{-n} \tag{1a}$$

$$y_{n+1} = y_n + \text{sign}(\omega_n) \cdot x_n \cdot 2^{-n} \tag{1b}$$

$$\omega_{n+1} = \omega_n - \text{sign}(\omega_n) \cdot \tan^{-1}(2^{-n}) \tag{1c}$$

for $0 \leq n \leq N - 1$, where N is the number of micro-rotations, and $\text{sign}(\omega_n) = -1$ if $\omega_n < 0$ and $\text{sign}(\omega_n) = 1$ otherwise. The initial value of $[x_0 \ y_0]$ is a two-dimensional vector to be rotated, and ω_0 is set to the rotation angle θ .

Figure 1a shows the structure of the conventional CORDIC circuit for recursive implementation of all micro-rotations whereas Figure 1b shows the structure of the cascaded CORDIC where possible locations of pipeline registers are marked with a thick solid lines. The barrel-shifters in Figure 1a can be completely eliminated in the cascaded CORDIC because the shift operations can be hardwired.

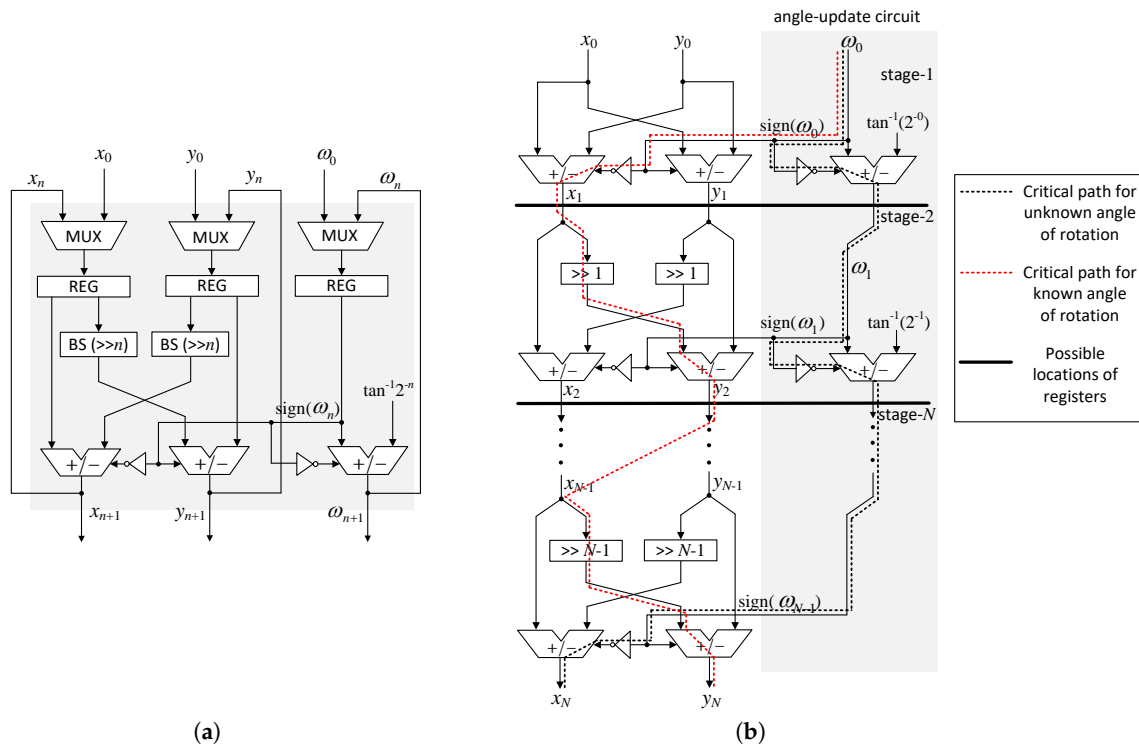


Figure 1. (a) A conventional recursive coordinate rotation digital computer (CORDIC) circuit. BS: barrel shifter; REG: register. (b) N -stage non-recursive cascaded CORDIC.

If the rotation angle is not known in advance, i.e., $\text{sign}(\omega_n)$ for $0 \leq n \leq N - 1$ is not given, a circuit for angle update according to Equation (1c) is required. The propagation delay of the fully-pipelined-cascaded CORDIC (FPCC), where all the thick solid lines between the CORDIC sections in Figure 1b are replaced with pipeline registers, can be represented as

$$T_{\text{FPCC}}(L) = T_{\text{INV}} + T_{\text{ADDSUB}}(L). \tag{2}$$

where T_{INV} and $T_{\text{ADDSUB}}(L)$ are propagation delays of an inverter and L -bit adder or subtractor, respectively. Sign extension to avoid overflow is not considered because the word-length is generally

retained during CORDIC operations. Assuming that a ripple carry adder (RCA) is used for the adder or subtractor, we have [25]

$$T_{\text{ADDSUB}}(L) = T_{\text{XOR}} + T_{\text{FAAC}} + (L - 2)T_{\text{FACC}} + T_{\text{FACS}} \tag{3}$$

where T_{XOR} is a delay of a 2-input XOR gate. T_{FAAC} , T_{FACC} , and T_{FACS} are delays of the 1-bit full adder (FA) from port input-A to port carry-out, from port carry-in to port carry-out, and from port carry-in to port sum, respectively. Note that the propagation delay of an adder or subtractor increases by T_{FACC} as the bit-width of the input increases by 1.

If all pipeline registers marked by thick solid lines in Figure 1b are removed, we can obtain a non-pipelined-cascaded CORDIC (NPCC), and the black-dotted line in Figure 1b becomes the critical path. Then, the propagation delay of an N -stage, L -bit NPCC $T_{\text{NPCC}}(N, L)$ can be generalized as follows:

$$T_{\text{NPCC}}(N, L) = N(T_{\text{INV}} + T_{\text{ADDSUB}}(L)) \tag{4}$$

where $(N - 1)(T_{\text{INV}} + T_{\text{ADDSUB}}(L))$ is taken to obtain $\text{sign}(\omega_1)$, $\text{sign}(\omega_2)$, ..., $\text{sign}(\omega_{N-1})$, and another $T_{\text{INV}} + T_{\text{ADDSUB}}(L)$ is the propagation delay of the inverter and adder or subtractor in the last stage.

3. Critical Path Analysis and Design of Cascaded CORDIC for Known Rotation Angles

3.1. Critical Path Analysis of Non-Pipelined Cascaded CORDIC

In this section, the precise critical path is analysed when signs for all the micro-rotations are known in advance assuming that the input rotation angle is known. Removing all pipeline registers and the angle-update circuit yields a non-pipelined cascaded CORDIC (NPCC) whose critical path is shown in the red-dotted line in Figure 1b. A detailed bit-level block diagram of an N -stage, L -bit NPCC is shown in Figure 2, where x_{nl} and y_{nl} are the l th bit of x_n and y_n , respectively, and the corresponding critical path is shown in the red-dotted line. Note that in Figure 2, the word-lengths of x_n and y_n for $0 \leq n \leq N$ are set to the number of stages such that $L = N$. To estimate the propagation delay of the NPCC, let us first find the propagation delay of the first stage assuming that NPCC has only one stage (shown in the first row in Figure 2). Note that the propagation delay of the first stage is the path from the input $\text{sign}(\omega_0)$ to the most significant bit (MSB) of the output $x_{1(L-1)}$, whose delay is equal to the sum of the propagation delay of the L -bit adder or subtractor in Equation (3) and the delay of an inverter. Then, the propagation delay of the first stage in an L -bit NPCC is

$$T_{\text{NPCC}}(1, L) = T_{\text{INV}} + T_{\text{ADDSUB}}(L) \tag{5}$$

where $T_{\text{NPCC}}(n, L)$ is the propagation delay of the n -th stage in an L -bit NPCC.

Now assume that one more stage is added for a total of two stages, but there are no pipeline registers between stages. One can see from Figure 1b that the input to the adder or subtractor on the right side in the second stage is obtained by right-shifting x_1 by 1. Therefore, addition or subtraction in the second stage can begin only if the second least significant bit (LSB) of x_1 , i.e., x_{11} , is available. Note that x_{11} is the LSB in the input of the adder or subtractor after the LSB x_{10} is truncated. Therefore, the propagation delay of a 2-stage NPCC is the sum of the delay from $\text{sign}(\omega_0)$ to x_{11} in the first stage and the delay from x_{11} to $y_{2(L-1)}$ in the second stage. The increased propagation delay due to inclusion of one more stage becomes $T_{\text{XOR}} + T_{\text{FAAC}} + T_{\text{FACS}}$, which is much lower than the delay of one L -bit RCA. Similarly, when one more stage is added to n -stage NPCC, yielding a total of $n + 1$ stages, the propagation delay increment Δ_n can be estimated as follows:

$$\Delta_n = T_{\text{XOR}} + T_{\text{FAAC}} + (n - 1)T_{\text{FACC}} + T_{\text{FACS}}, \tag{6}$$

for $1 \leq n < N$, and

$$T_{NPCC}(n + 1, L) = T_{NPCC}(n, L) + \Delta_n. \tag{7}$$

It should be noted that the increased propagation delay does not depend on the input word-length but rather on the number of shifts in the CORDIC unit when separate pipeline stages are merged, which is much less than the delay of $T_{ADDSUB}(L)$, especially in the initial CORDIC stages. The propagation delay of the N -stage NPCC is

$$T_{NPCC}(N, L) = T_{NPCC}(1, L) + \sum_{n=1}^{N-1} \Delta_n. \tag{8}$$

Substituting Equations (3), (5), and (6) into Equation (8) yields

$$T_{NPCC}(N, L) = T_{INV} + N(T_{FAAC} + T_{XOR} + T_{FACS}) + ((N^2 - 3N)/2 + L - 1)T_{FACC}. \tag{9}$$

Equation (9) shows that the propagation delay of the NPCC can be estimated for any given values of N and L if T_{INV} , T_{FAAC} , T_{XOR} , T_{FACS} , and T_{FACC} are approximated. Table 1 lists the estimated propagation delays of the N -stage, 16-bit NPCC for $1 \leq N \leq 16$ and the actual propagation delays obtained from the synthesis results using the TSMC 90 nm CMOS library [26]. One can see from the second column in Table 1 that the delay increment Δ_N increases with N because more shifts are performed in later stages.

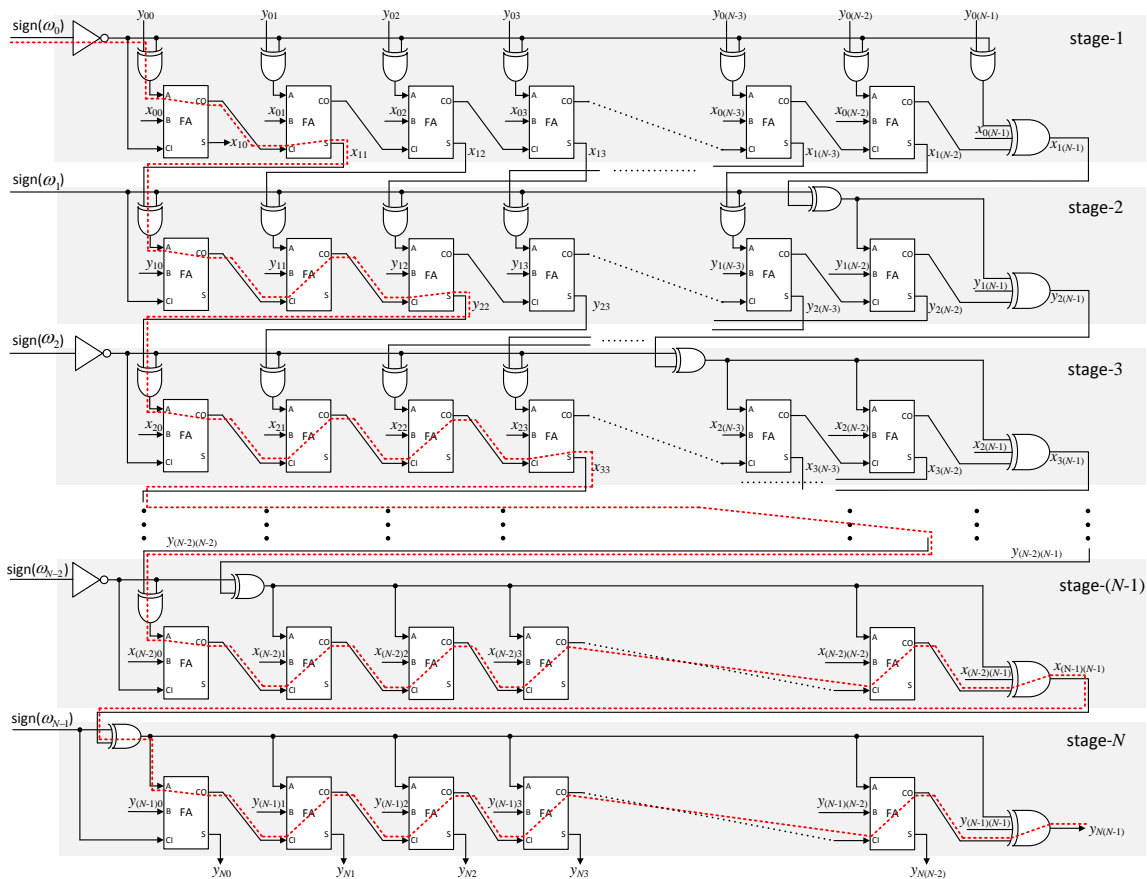


Figure 2. Detailed bit-level block diagram and critical path of an N -stage non-pipelined cascaded CORDIC without an angle update circuit. The word length L is assumed to be equal to the number of stages N .

Table 1. Propagation delay estimates (ns) of an N -stage, 16-bit non-pipelined-cascaded CORDIC (NPCC), and corresponding synthesis results obtained using the TSMC 90 nm CMOS library.

N	Δ_N	T_{NPCC}	Syn. Results
1	0.45	1.74	1.70
2	0.53	2.19	2.09
3	0.62	2.72	2.59
4	0.70	3.34	3.20
5	0.79	4.05	3.92
6	0.87	4.84	4.75
7	0.96	5.71	5.69
8	1.04	6.67	6.74
9	1.13	7.72	7.76
10	1.21	8.85	8.88
11	1.30	10.06	10.09
12	1.38	11.36	11.40
13	1.47	12.75	12.80
14	1.55	14.22	14.29
15	1.64	15.77	15.88
16	.	17.41	17.56

$T_{INV} = 0.10$ ns, $T_{XOR} = 0.12$ ns, $T_{FAAC} = 0.18$ ns, $T_{FACC} = 0.08$ ns, and $T_{FACS} = 0.15$ ns were set for the simulation by [26], but they can be adjusted based on the temperature and output load capacitance.

3.2. Critical Path Analysis of Pipelined Cascaded CORDIC

All stages in a fully-pipelined cascaded CORDIC have the same critical path, which is from the inverter to the adder or subtractor in a stage, and its throughput is $1/T_{FPCC}(L)$ in Equation (2). When such a high throughput rate is not required, some pipeline registers can be removed in order to merge stages. In this section, we analyse the propagation delay of a partially-pipelined-cascaded CORDIC (PPCC), where some of the pipeline stages are merged. Let us denote $T_{PPCC}(L)(n, m)$ as the propagation delay from the n -th stage to the m -th stage (assuming that the pipeline registers are located before the n -th stage and after the m -th stage) in an L -bit PPCC, and there are no other registers between them. Then, $T_{PPCC}(L)(n, m)$ is determined from the longest path between the input of the n -th stage (x_{n-1} or y_{n-1}) and the output of the m -th stage (x_m or y_m). If $n = m$, the registers are located before and after the n -th stage, thus

$$T_{PPCC}(L)(n, n) = T_{FPCC}(L), \tag{10}$$

for $1 \leq n \leq N$. Also, we have

$$T_{PPCC}(L)(n, n + 1) = T_{PPCC}(L)(n, n) + \Delta_n \tag{11}$$

where Δ_n is defined in Equation (6). Therefore,

$$T_{PPCC}(L)(n, m) = T_{PPCC}(L)(n, n) + \sum_{k=n}^{m-1} \Delta_k. \tag{12}$$

Substituting Equations (6) and (10) into Equation (12) yields the propagation delay from the n -th stage to the m -th stage in a PPCC:

$$T_{PPCC}(L)(n, m) = T_{INV} + (m - n + 1)(T_{FAAC} + T_{XOR} + T_{FACS}) + ((m^2 - 3m - n^2 + 3n + 2L - 4)/2)T_{FACC}. \tag{13}$$

Let us denote the propagation delay of the K -stage in an L -bit PPCC involving N micro-rotations and $(K - 1)$ pairs of pipeline registers as $T_{PPCC}(N, K, L)$. Specifically, if the registers are located before the n_1 -th stage, the n_2 -th stage, . . . , and the n_{K-1} -th stage, $T_{PPCC}(N, K, L)$ can be determined from

$$T_{PPCC}(N, K, L) = \max \{ T_{PPCC}(L)(1, n_1 - 1), T_{PPCC}(L)(n_1, n_2 - 1), T_{PPCC}(L)(n_2, n_3 - 1), \dots, T_{PPCC}(L)(n_{K-1}, N) \}. \tag{14}$$

3.3. Algorithm Design for a Pipelined Cascaded CORDIC

Based on the above analysis, we propose an algorithm for minimizing the number of pipeline stages and the locations of pipeline registers in a cascaded CORDIC. Figure 3 shows a flowchart of the proposed search algorithm. The goal of the algorithm is to minimize the number of pipeline stages in order to minimize pipeline overhead without violating the timing constraint per stage. For a given timing constraint T , assume that

$$T_{PPCC}(L)(n, n) < T, \quad \text{for } 1 \leq n \leq N, \tag{15}$$

otherwise, the PPCC can never meet the timing requirement. It is assumed that the values of T_{INV} , T_{FAAC} , T_{XOR} , T_{FACS} , and T_{FACC} are known, and the values of n , m , and k are initialized. To search for the first location of the pipeline registers, $T_{PPCC}(L)(1, 2)$ is estimated according to Equation (13). If the value of $T_{PPCC}(L)(1, 2)$ is less than T , we can infer that pipeline registers are not required between the first and second stages. Then, m is increased by 1 to test whether the registers should be placed before the third stage by calculating $T_{PPCC}(L)(1, 3)$. In this manner, the value of m is increased by 1 until $T_{PPCC}(L)(1, m)$ is larger than the timing constraint. If the value of $T_{PPCC}(L)(1, m)$ is larger than the value of T , the pipeline registers should be placed after the $(m - 1)$ -th stage. Then, the location of the first pipeline registers $(m - 1)$ is stored in $P(0)$, and the algorithm continues to search for the next location of pipeline registers $P(1)$. Because the pipeline registers are located before the m -th stage, the initial position of propagation delay n is reset to m for the next search, and the value of m is increased until the next location of pipeline register is found or m reaches the end of the stage. When the algorithm terminates, the location of pipeline registers can be retrieved from $P(k)$, and the total number of pipeline stages K is taken to be $(k + 1)$. Pseudo-code of the algorithm is given in Algorithm 1.

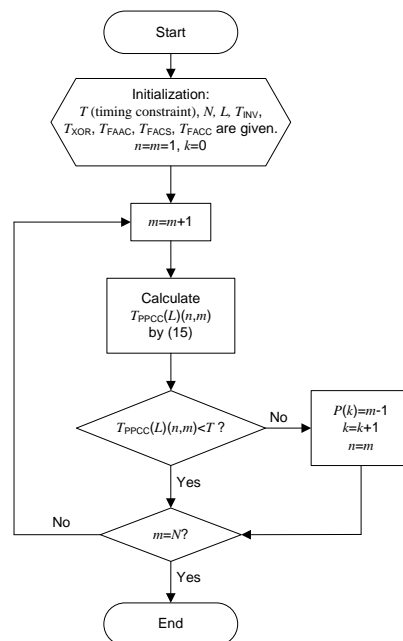


Figure 3. The algorithm used to determine number of pipeline stages and to choose locations of pipeline registers in the cascaded CORDIC.

Algorithm 1 Searches the locations of pipeline registers

```

1: set  $T, N, L$ ;
2: obtain  $T_{INV}, T_{XOR}, T_{FAAC}, T_{FACS}, T_{FACC}$ ;
3:  $n = 1; m = 1; k = 0$ ;
4: do
5:    $m = m + 1$ ;
6:   if ( $T_{PPCC}(L)(n, m) \geq T$ )
7:      $P(k) = m - 1$ ;
8:      $k = k + 1$ ;
9:      $n = m$ ;
10:  end if
11: while ( $m < N$ )
12: end while

```

In Table 2, we show design examples of a cascaded CORDIC obtained with the proposed critical path analysis and design algorithm when the timing constraint is defined as a maximum usable frequency (MUF) ranging from 100 MHz to 300 MHz and when $N = L = 16$. The minimum clock period T in the third column of Table 2 is the reciprocal of the corresponding MUF. Six examples with different constraints are designed with the TSMC 90 nm standard CMOS libraries [26]. In the example shown in Design-1, NPCC is used when any timing constraint is not given, and its estimated propagation delay is 17.41 ns. If higher throughput is required, a pipelined-cascaded CORDIC can be used with a few pipeline stages. In Design-2, only one pair of pipeline registers is sufficient to accommodate the timing constraint. It should be noted that the first pipeline stage in Design-3 has seven CORDIC units, whereas the second pipeline stage has only five CORDIC units because Δ_n is larger at later stages due to larger shifts. More pipeline registers are used as the minimum required clock frequency increases, as shown in Table 2. In Design-6 (which requires minimum clock frequency of 300 MHz), the PPCC requires seven pairs of pipeline registers in order to have eight pipeline stages.

Table 2. Design examples of a cascaded CORDIC obtained using the TSMC 90 nm standard CMOS library ($N = L = 16$).

Design	Clock (MHz)	T (ns)	Location of Pipeline Registers							Estimated Delay (ns)
			$P(0)$	$P(1)$	$P(2)$	$P(3)$	$P(4)$	$P(5)$	$P(6)$	
Design-1	×	×	×	×	×	×	×	×	×	$T_{NPCC}(16, 16) = 17.41$
Design-2	100	10.00	10	×	×	×	×	×	×	$T_{PPCC}(16)(11, 16) = 9.09$
Design-3	150	6.66	7	12	×	×	×	×	×	$T_{PPCC}(16)(8, 12) = 6.43$
Design-4	200	5.00	6	10	13	×	×	×	×	$T_{PPCC}(16)(14, 16) = 4.93$
Design-5	250	4.00	4	7	10	12	14	×	×	$T_{PPCC}(16)(8, 10) = 3.91$
Design-6	300	3.33	3	6	8	10	12	14	15	$T_{PPCC}(16)(4, 6) = 3.23$

$T_{INV} = 0.10$ ns, $T_{XOR} = 0.12$ ns, $T_{FAAC} = 0.18$ ns, $T_{FACC} = 0.08$ ns, and $T_{FACS} = 0.15$ ns were set in the simulation by [26], but they can be adjusted based on the temperature and output load capacitance.

4. A Recursive Cascaded CORDIC

In this Section, we propose a recursive cascaded CORDIC design, namely a pipelined-recursive-cascaded CORDIC (PRCC). Figure 4 shows the structure of a 2-stage PRCC that performs N successive micro-rotations using two CORDIC units. When N is even, the first $N/2$ micro-rotations out of N micro-rotations are performed during the first stage, whereas the remaining $N/2$ micro-rotations are performed during the second stage. A pair of 2:1 MUXes in the first stage selects the initial input values x_0 and y_0 during the first clock cycle of the period of the first $N/2$ clock cycles, and it selects outputs from two adders or subtractors in the first stage during the subsequent $N/2 - 1$ clock cycles. The first stage is responsible only for the first $N/2$ cycles of micro-rotations and passes the intermediate results

$x_{N/2}$ and $y_{N/2}$ to the second stage. Another pair of inputs are taken and fed in parallel to the MUXes in every $N/2$ cycles. A pair of 2:1 MUXes in the second stage selects $x_{N/2}$ and $y_{N/2}$ from the first stage every $N/2$ cycles, and the second stage performs the remaining $N/2$ micro-rotations. PRCC can also be implemented with more than two CORDIC units. If N is represented as a multiple of K , where $K > 2$, we can have a K -stage PRCC, where each stage is responsible for (N/K) micro-rotations, and each stage receives a pair of inputs from the previous stage every (N/K) cycles.

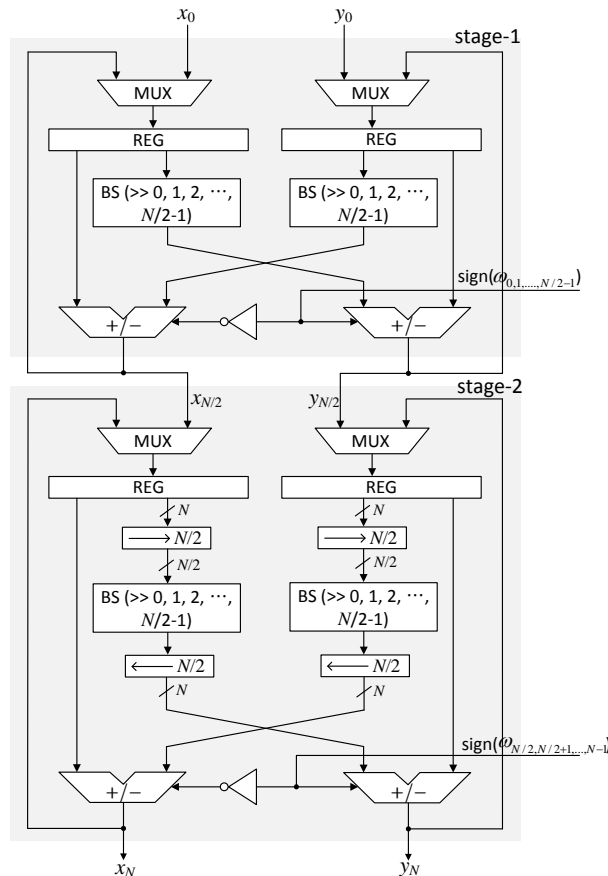


Figure 4. Structure of a 2-stage pipelined recursive cascaded CORDIC. $\rightarrow N/2$ indicates a right-shift by $(N/2)$ bit locations. $\leftarrow N/2$ indicates a sign-extension by $(N/2)$ -bits.

The hardware-complexity of the barrel-shifter in a PRCC can be effectively reduced with a simple hardwired pre-shifting scheme [7], as shown in the second stage of Figure 4. Only the $(N/2)$ most significant bits of the input words from the registers can be loaded to the barrel-shifters, because $N/2$ is the minimum number of shifts in the second stage and the $N/2$ less significant bits would become truncated during shifting. Therefore, the barrel-shifter must implement a maximum of $N/2$ shifts. The output from the barrel-shifters are loaded into the $(N/2)$ LSBs of the adder or subtractor, and the $N/2$ MSBs of the corresponding operand are sign-extended. The barrel-shifter in the second stage can be implemented using $N/2 \cdot \lceil \log_2(N/2) \rceil$ number of MUXes, whereas $N \cdot \lceil \log_2(N) \rceil$ MUXes are required without using hardwired pre-shifting. Therefore, the area complexity of a PRCC can be reduced by implementing hardwired pre-shifting.

The propagation delay of the barrel-shifter could also be decreased by hardwired pre-shifting. As the maximum number of shifts in the first and second stages are equal, as shown in Figure 4, each stage has the same propagation delay $T_{MUX} \cdot \lceil \log_2(N/2) \rceil$, where T_{MUX} is the propagation delay of the 2:1 MUX. Therefore, the critical path of a PRCC is determined by any of its stages. However, the critical path of a PRCC is less than that of a conventional CORDIC because the maximum number of shifts

with the barrel-shifters in a PRCC is less than that in a conventional CORDIC. The propagation delay of the K -stage, L -bit PRCC with N micro-rotations is

$$T_{PRCC}(N, K, L) = (\lceil \log_2(N/K) \rceil + 1)T_{MUX} + T_{ADDSUB}(L). \tag{16}$$

Note that $\lceil \log_2(N/K) \rceil T_{MUX}$ is the delay of a barrel-shifter and another T_{MUX} is the delay of a 2:1 MUX used for input selection.

5. Design Examples and Results

The hardware and time complexities of the proposed designs and conventional designs for known rotation angles are listed in Table 3. The proposed K -stage PPCC is compared with an NPCC and FPCC [21], which are two opposing CORDIC versions in terms of pipeline strategy in a cascaded CORDIC. The proposed K -stage PRCC is also compared with a conventional CORDIC [1] computed through recursive computation of one stage and opposing non-recursive cascaded CORDIC computed through N stages. The propagation delay in a conventional CORDIC is the sum of delays of a 2:1 MUX, barrel-shifter, and adder or subtractor. FPCC has the shortest critical path, which involves an inverter and an adder or subtractor. The FPCC, NPCC, and PPCC produce an output sample during every cycle when they are used in the pipelined CORDIC applications, whereas the conventional CORDIC and K -stage PRCC produce the output sample every N and N/K cycles, respectively. The K -stage PRCC has K CORDIC units, thus it involves a factor K more adders or subtractors, barrel-shifters, registers, and MUXes than a conventional CORDIC. However, the area of a K -stage PRCC is less than a factor K greater than that of a conventional CORDIC due to its hardwired pre-shifting. The propagation delay of a PRCC can be reduced to be less than that of a conventional CORDIC.

Table 3. Comparison of hardware and time complexities from different CORDIC architectures with N micro-rotations and L -bit inputs.

Design	Propagation Delay	Throughput	Number of Hardware Elements			
			ADDSUB	BS	REG	MUX
Conventional [1]	$T_{MUX} + T_{BS} + T_{ADDSUB}(L)$	$1/N$	2	2	2	2
FPCC [21]	$T_{FPCC}(L)$ in Equation (2)	1	$2N$	0	$2N$	0
NPCC	$T_{NPCC}(N, L)$ in Equation (9)	1	$2N$	0	0	0
K -Stage PPCC	$T_{PPCC}(N, K, L)$ in Equation (14)	1	$2N$	0	$2(K - 1)$	0
K -Stage PRCC	$T_{PRCC}(N, K, L)$ in Equation (16)	K/N	$2K$	$2K$	$2K$	$2K$

Conventional: conventional CORDIC, MUX: 2:1 multiplexor, BS: barrel-shifter, ADDSUB: adder or subtractor, and REG: register.

We coded Designs-1, 2, 3, 4, and 6 in the design examples in Table 2 in VHDL and synthesized those using the Synopsys Design Compiler with the TSMC 90 nm general purpose standard CMOS library [26]. A conventional CORDIC [1], 2, 4, and 8-stage PRCCs, and an FPCC [21] were also synthesized and compared for known rotation angles. The values of L and N were 16. The maximum propagation delay, throughput per second (TPS), latency required to obtain the first output sample, area, power consumption at 50 MHz operating frequency, energy consumed per sample (EPS) required to produce a 50 MHz clock output, and area-delay product (ADP) are listed in Table 4.

The conventional CORDIC has the smallest area of $2092 \mu\text{m}^2$ because it involves only one CORDIC unit with throughput rate of 26.1 Mega samples per second (MSPS). It is shown that PRCCs have lower delays than a conventional CORDIC due to the hardwired pre-shifting. Specifically, a 2-stage PRCC occupies a factor 1.6 greater area than a conventional CORDIC, but it offers nearly double the throughput in the pipeline CORDIC application, 7.9% less EPS, and 20.7% less ADP. Similarly, the 4-stage and 8-stage PRCCs provide factors 4.3 and 8.9 larger throughput, 11.7% and 18.3% less EPS, and 32.3% and 40.4% less ADP compared to the conventional design, respectively. FPCC can produce an output during every cycle at 549 MHz maximum operating frequency. However, in applications that do not require such high throughput, NPCC or PPCC can be alternative architectural options by

removing unnecessary pipeline registers. If the throughput rate of less than 60 MSPS is acceptable, NPCC becomes the one of the best options as it requires 38.8% less area and uses 11.9% less EPS compared to the fully-pipelined design. 2, 3, 4, and 8-stage PPCCs produce 109 MSPS, 155 MSPS, 202 MSPS, and 310 MSPS, but require 36.3%, 33.7%, 31.1%, and 20.7% less area with 24.1%, 29.6%, 29.0%, and 23.4% less EPS than FPCC, respectively. The 4 and 8-stage PPCCs offer 45.8% to 41.5% savings in EPS and 28.7% and 46.6% savings in ADP over a conventional CORDIC. One should note that the propagation delays of PPCCs obtained by synthesis closely match the estimated propagation delays shown in Table 2.

Table 4. Performance comparison of the conventional and cascaded CORDIC algorithms based on synthesis results for known rotation angles ($N = L = 16$).

Design	Delay (ns)	TPS (ns)	Latency (MSPS)	Area (μm^2)	Power (mW)	EPS (mW·ns)	ADP ($\mu\text{m}^2\cdot\text{ns}$)
Conventional [1]	2.39	26.15	38.24	2092	0.04	13.46	79,998
2-Stage PRCC	2.30	54.34	36.80	3444	0.07	12.39	63,369
4-Stage PRCC	2.22	112.61	35.52	6090	0.14	11.88	54,079
8-Stage PRCC	2.14	233.64	34.24	11,138	0.27	10.98	47,670
NPCC (Design-1)	17.57	56.91	17.57	10,211	0.45	9.05	179,407
2-Stage PPCC (Design-2)	9.15	109.28	18.30	10,644	0.38	7.79	97,392
3-Stage PPCC (Design-3)	6.45	155.03	19.35	11,077	0.36	7.23	71,446
4-Stage PPCC (Design-4)	4.95	202.02	19.80	11,511	0.36	7.29	56,979
8-Stage PPCC (Design-6)	3.22	310.55	25.76	13,244	0.39	7.87	42,645
FPCC [21]	1.82	549.45	29.12	16,710	0.51	10.28	30,412

Conventional: conventional CORDIC, TPS: throughput per second, Power: power consumption at 50 MHz clock frequency, EPS: energy consumed per sample at 50 MHz clock frequency, and ADP: area-delay product. Design- n is listed in Table 2.

We designed NPCC, PPCC, and FPCC for unknown rotation angles. The proposed designs for different number of pipeline stages with different timing constraints were synthesized, and Table 5 lists the synthesis results. Note that 3, 4, and 6 stages are sufficient to produce greater than 100 MSPS, 150 MSPS, and 200 MSPS TPS, which requires 31.3%, 28.1%, and 24.1% less area and 19.0%, 25.2%, and 25.6% less EPS than an FPCC, respectively. From Table 5, we know that remarkable savings in terms of area and EPS can also be obtained over fully-pipelined designs, even for unknown rotation angles.

Table 5. Performance comparison of the conventional and cascaded CORDIC algorithms based on synthesis results for unknown rotation angles ($N = L = 16$).

Design	Delay (ns)	TPS (ns)	Latency (MSPS)	Area (μm^2)	Power (mW)	EPS (mW·ns)	ADP ($\mu\text{m}^2\cdot\text{ns}$)
NPCC	20.29	49.28	20.29	18,161	0.37	7.48	368,486
3-Stage PPCC	9.88	101.21	29.64	19,305	0.31	6.34	190,733
4-Stage PPCC	6.32	158.22	25.28	20,198	0.29	5.86	127,651
6-Stage PPCC	4.52	221.23	27.12	21,336	0.29	5.83	96,438
FPCC [21]	1.91	523.56	30.56	28,111	0.39	7.84	53,692

TPS: throughput per second, Power: power consumption at 50 MHz clock frequency, EPS: energy consumed per sample at 50 MHz clock frequency, and ADP: area-delay product.

6. Summary and Conclusions

A conventional CORDIC is inherently sequential and involves large latency. It offers low throughput rate due to its recursive implementation of micro-rotations. In contrast, a fully-pipelined non-recursive cascaded CORDIC provides very high throughput rate at the cost of large area complexity. However, such high throughput is not required in many applications. For example when CORDIC is used for implementing the complex butterfly operation for FFT calculations, the throughput rate of CORDIC

must match the throughput requirement of the application and the type of FFT implementation. On the other hand, we see that the critical path does not increase substantially when some of the pipeline stages are removed from a fully-pipelined cascaded CORDIC. Therefore, we have explored other design choices for a CORDIC with varying operating frequency, throughput rate, latency, and area complexity. In this paper, we present a precise estimate of the propagation delays in CORDIC circuits to determine the critical paths in the pipelined and non-pipelined recursive and non-recursive CORDIC architectures. We have shown that the propagation delay does not increase significantly and does not depend on the input word length when adjacent pipeline stages are merged. Instead, the propagation delay depends on the number of shifts in the CORDIC unit. Therefore, more initial stages in CORDIC can be merged to form a single pipeline stage compared to the later stages. We proposed an algorithm to search for the locations of pipeline registers in order to minimize the number of pipeline stages and determine the desired critical path for a given timing constraint. We have also proposed a hybrid cascaded CORDIC and recursive CORDIC to increase throughput and save energy per sample. We have shown that PPCC requires less area with lower EPS than an FPCC, and a PRCC operates with less EPS and has lower ADP compared to the conventional recursive design.

Author Contributions: Conceptualization, P.K.M. and S.Y.P.; Methodology, P.K.M. and S.Y.P.; Validation, S.Y.P.; Writing—original draft preparation, S.Y.P.; Writing—review and editing, P.K.M. and S.Y.P.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (grant number: 2016R1D1A1B03933315).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations for various CORDICs are used in this manuscript:

CORDIC	COordinate Rotation Digital Computer
FPCC	Fully-Pipelined-Cascaded CORDIC
PRCC	Pipelined-Recursive-Cascaded CORDIC
NPCC	Non-Pipelined-Cascaded CORDIC
PPCC	Partially-Pipelined-Cascaded CORDIC

References

1. Volder, J.E. The CORDIC trigonometric computing technique. *IRE Trans. Electron. Comput.* **1959**, *EC-8*, 330–334. [[CrossRef](#)]
2. Walther, J.S. A unified algorithm for elementary functions. In Proceedings of the 38th Spring Joint Computer Conference, Atlantic City, NJ, USA, 18–20 May 1971; pp. 379–385.
3. Hu, Y.H. CORDIC-based VLSI architectures for digital signal processing. *IEEE Signal Process. Mag.* **1992**, *9*, 16–35. [[CrossRef](#)]
4. Meher, P.L.; Valls, J.; Juang, T.-B.; Sridharan, K.; Maharatna, K. 50 years of CORDIC: Algorithms, architectures and applications. *IEEE Trans. Circ. Syst. I Regul. Pap.* **2009**, *56*, 1893–1907. [[CrossRef](#)]
5. Hu, Y.H.; Naganathan, S. An angle recoding method for CORDIC algorithm implementation. *IEEE Trans. Comput.* **1993**, *42*, 99–102. [[CrossRef](#)]
6. Hu, Y.H.; Chern, H.H.M. A novel implementation of CORDIC algorithm using backward angle recoding (BAR). *IEEE Trans. Comput.* **1996**, *45*, 1370–1378.
7. Meher, P.K.; Park, S.Y. CORDIC Designs for Fixed Angle of Rotation. *IEEE Trans. Very Large Scale Integr. Syst.* **2013**, *21*, 217–228. [[CrossRef](#)]
8. Wu, C.S.; Wu, A.Y.; Lin, C.H. A high-performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes. *IEEE Trans. Circuits Syst. II Anal. Digit. Signal Process.* **2003**, *50*, 589–601.
9. Juang, T.B.; Hsiao, S.F.; Tsai, M.Y. Para-CORDIC: Parallel CORDIC rotation algorithm. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2004**, *51*, 1515–1524. [[CrossRef](#)]

10. Rodrigues, T.K.; Swartzlander, E.E. Adaptive CORDIC: Using parallel angle recoding to accelerate CORDIC rotations. In Proceedings of the Fortieth Asilomar Conference on Signals, Systems and Computers, ACSSC'06, Pacific Grove, CA, USA, 29 October–1 November 2006; pp. 323–327.
11. Maharatna, K.; Banerjee, S.; Grass, E.; Krstic, M.; Troya, A. Modified virtually scaling free adaptive CORDIC rotator algorithm and architecture. *IEEE Trans. Circuits Syst. Video Technol.* **2005**, *15*, 1463–1474. [[CrossRef](#)]
12. Kang, C.Y. Digit-pipelined direct digital frequency synthesis based on differential CORDIC. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2006**, *53*, 1035–1044. [[CrossRef](#)]
13. Graham, P.; Nelson, B. Reconfigurable Processors for High-Performance, Embedded Digital Signal Processing. In *Field Programmable Logic and Applications*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 1–10.
14. Tutueva, A.V.; Butusov, D.N.; Pesterev, D.O.; Belkin, D.A.; Ryzhov, N.G. Novel normalization technique for chaotic Pseudo-random number generators based on semi-implicit ODE solvers. In Proceedings of the 2017 International Conference “Quality Management, Transport and Information Security, Information Technologies” (IT QM IS), St. Petersburg, Russia, 24–30 September 2017; pp. 292–295.
15. Cavallaro, J.R.; Luk, F.T. CORDIC arithmetic for a SVD processor. *J. Parallel Distrib. Comput.* **1988**, *5*, 271–290. [[CrossRef](#)]
16. Hwang, K.; Faye, A. *Computer Architecture and Parallel Processing*; McGraw-Hill College: New York, NY, USA, 1984.
17. Parhi, K.K. *VLSI Digital Signal Processing Systems: Design and Implementation*; John Wiley & Sons, Inc.: New York, NY, USA, 1999.
18. Kung, S.Y. *VLSI Array Processors*; Prentice Hall: Upper Saddle River, NJ, USA, 1988.
19. Chen, J.; Liu, K. A complete pipelined parallel CORDIC architecture for motion estimation. *IEEE Trans. Circuits Syst. II Anal. Digit. Signal Process.* **1998**, *45*, 653–660. [[CrossRef](#)]
20. Hsiao, S.F.; Hu, Y.H.; Juang, T.B. A memory-efficient and high-speed sine/cosine generator based on parallel CORDIC rotations. *IEEE Signal Process. Lett.* **2004**, *11*, 152–155. [[CrossRef](#)]
21. Deprettere, E.; Dewilde, P.; Udo, R. Pipelined CORDIC architectures for fast VLSI filtering and array processing. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '84, San Diego, CA, USA, 19–21 March 1984; Volume 9, pp. 250–253.
22. Jun, M.; Parhi, K.K.; Hekstra, G.J.; Deprettere, E.F. Efficient implementations of pipelined CORDIC based IIR digital filters using fast orthonormal μ -rotations. *IEEE Trans. Signal Process.* **2000**, *48*, 2712–2716. [[CrossRef](#)]
23. Aggarwal, S.; Khare, K. Low complexity VLSI implementation of CORDIC-based exponent calculation for neural networks. *Int. J. Electron.* **2012**, *99*, 1471–1488. [[CrossRef](#)]
24. Garcia, E.I.; Cumplido, R.; Arias, M. Pipelined CORDIC design on FPGA for a digital sine and cosine waves generator. In Proceedings of the International Conference on Electrical and Electronics Engineering, ICEEE'06, Veracruz, Mexico, 6–8 September 2006; pp. 1–4.
25. Meher, P.K.; Park, S.Y. Critical-Path Analysis and Low-Complexity Implementation of LMS Adaptive Algorithm. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2014**, *61*, 778–788. [[CrossRef](#)]
26. TSMC 90nm General-Purpose CMOS Standard Cell Libraries—tcbn90ghptc. Available online: <https://www.tsmc.com/> (accessed on 29 March 2019).



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).