Complexity Reductions in Unrolled CORDIC Architectures

Peter Nilsson, Member IEEE,
Department of Electrical and Information Technology,
Faculty of Engineering, Lund University,
Lund, Sweden,
Peter.Nilsson@eit.lth.se

Abstract—This paper shows a novel methodology to reduce the complexity in unrolled CORDIC architectures. The methodology is based on eliminating the CORDIC stages starting from the first stage. As an example, a six stage CORDIC is used but the methodology is applicable on CORDICs with an arbitrary number of stages. The paper shows that the complexity can be reduced by 25%.

I. INTRODUCTION

To compute non-linear functions is in most cases a demanding task to do in hardware. Look-up tables are useful if the precision is low but the size of the table grows exponentially, which make them unfeasible for hardware when the precision is high. Recently, a methodology using a sum of bit products for cosine and sine generation is presented [1]. Another, novel methodology is to use parabolic functions to compute unary functions. The parabolic methodology has the advantage that the delay is short [2][3]. This paper focus on the CORDIC algorithm [4], which benefits from that it is multiplier less, i.e. it is based on additions and subtractions only. Traditionally, iterative CORDICs [5] have been used due to their small area. Today, the area is less important compared to other parameters such as the power consumption. It is thus feasible to use unrolled CORDICs. However, it is still important to reduce the complexity since it can reduce the power consumption as well. This paper shows a novel methodology to reduce the complexity in unrolled CORDIC architectures.

II. THE CORDIC ALGORITHM

The CORDIC algorithm is based on vector rotations to find an approximation of a non-linear function. The main advantage with the algorithm is that it is multiplier less. It uses additions and subtractions only. The CORDIC is built with a number of stages. By increasing the number of stages the accuracy is improved. Fig. 1 shows an example of three vector rotations, corresponding to a three stage CORDIC. The dotted vector is the vector with an angle $\alpha$ of which, in this case, the approximate cosine and sine values are searched for.

There are many ways to choose where to start the rotations. In this paper, the starting point is chosen to be at the coordinates $(x = 1/R, y = 0)$, where $R$ is the last vector length if we had started at the unit circle. A starting point at $1/R$ will thus result in that we come as close to the unit circle as possible when the last stage is computed. $R$ is a fixed coefficient for the chosen number of CORDIC stages. It can thus be hardware wired in the implementation. When the last rotation is done, the coordinates give the approximate cosine and sine values for the input angle, i.e. $(x_3, y_3)$ in the figure.

III. THE ORIGINAL UNROLLED CORDIC

A six stage CORDIC is chosen to demonstrate the methodology. However, the methodology is valid for any number of stages. Fig. 2 shows the architecture of the six stage CORDIC, see at the end of the paper. In the six adders at the top, the remaining angle is computed for each stage.
The input variable \( \alpha \) is the angle for the cosine and sine that is searched for. At the top, the fixed coefficient angle values for each rotation are provided. These are added or subtracted from \( \alpha \) in each stage. The coefficients are in this demonstration, chosen to be 8-bit wide. The coefficients can be chosen with other bit widths, but it is not important in this particular case when the complexity reduction methodology is to be demonstrated. They are fixed coefficients but they can be stored in a ROM as well.

In the middle and the lower adder rows the approximation of the cosine and sine value is computed and provided to the right. The initial vector value with \( x \)-axis and \( y \)-axis coordinates is provided to the left, here \((x = 1/R, y = 0)\). In each stage new vectors are determined, in order to converge towards the vector that approximates the vector that represents the angle \( \alpha \). In each vector rotation stage there is a crosswise addition or subtraction of the vector coordinates. The vector rotations depend on the sign bits (\( sgn \)), in each stage of the top row of adders. That is, the sign bits determine if it should be an addition or subtraction. There are also divisions of the vector coordinates by a factor corresponding to \( 2^k \) where \( k \) is the integers \( \{1, 2, 4, 8, 16, 32\} \). That is, division by 1, 2, 4, 8, 16, and 32. That is, the right shifts as discussed in section II, which are done by shifting the busses between the stages. There is thus no extra cost in hardware for the divisions.

### IV. THE REDUCED COMPLEXITY CORDIC

To reduce the complexity, stages in the architecture in Fig. 2 can be eliminated. Starting from the left, the first stage can be removed directly since the \( x \) and \( y \) values are fixed. The methodology is valid for any chosen input vector. Here in this example, the coordinates \( x = 1/R \) and \( y = 0 \) is chosen, to show the complexity reduction methodology as discussed in section II. That means that the output from the first stage is \( x \) in both the middle and the lower adder row, as shown in 1.

\[
\begin{align*}
y_1 &= x \\
x_1 &= x
\end{align*}
\]

(1)

The stages are numbered from 1 to 6. In the second stage the coordinates will be as shown in 2.

\[
\begin{align*}
y_2 &= y_1 - \frac{x_1}{2} = \frac{x}{2} \\
x_2 &= x_1 + \frac{y_1}{2} = \frac{3x}{2}
\end{align*}
\]

(2)

if the sign bit from the first stage is positive. Here we see the first stage input coordinates added/subtracted with the crosswise one step shifted coordinates. If the sign bit is negative the corresponding vector coordinates will be according to 3.

\[
\begin{align*}
y_2 &= y_1 + \frac{x_1}{2} = \frac{3x}{2} \\
x_2 &= x_1 - \frac{y_1}{2} = \frac{x}{2}
\end{align*}
\]

(3)

The result will thus have the same values in the both cases but they are crosswise interleaved between the coordinates. To eliminate the second stage two MUXes can be used as shown in Fig. 3.

Figure 3. MUXes for eliminating the second stage

if \( sgn_1 = 0, sgn_2 = 0 \)

\[
\begin{align*}
y_3 &= y_2 + \frac{x_2}{4} = \frac{3x}{2} + \frac{x}{8} = \frac{13x}{8} \\
x_3 &= x_2 - \frac{y_2}{4} = \frac{x}{2} - \frac{3x}{8} = \frac{x}{8}
\end{align*}
\]

if \( sgn_1 = 1, sgn_2 = 0 \)

\[
\begin{align*}
y_3 &= y_2 + \frac{x_2}{4} = \frac{x}{2} + \frac{3x}{8} = \frac{7x}{8} \\
x_3 &= x_2 - \frac{y_2}{4} = \frac{3x}{2} - \frac{x}{8} = \frac{11x}{8}
\end{align*}
\]

if \( sgn_1 = 0, sgn_2 = 1 \)

\[
\begin{align*}
y_3 &= y_2 - \frac{x_2}{4} = \frac{3x}{2} - \frac{x}{8} = \frac{11x}{8} \\
x_3 &= x_2 + \frac{y_2}{4} = \frac{x}{2} + \frac{3x}{8} = \frac{7x}{8}
\end{align*}
\]

if \( sgn_1 = 1, sgn_2 = 1 \)

\[
\begin{align*}
y_3 &= y_2 - \frac{x_2}{4} = \frac{x}{2} - \frac{3x}{8} = \frac{x}{8} \\
x_3 &= x_2 + \frac{y_2}{4} = \frac{3x}{2} + \frac{x}{8} = \frac{13x}{8}
\end{align*}
\]

When it comes to the third stage, the \( x_3 \) and \( y_3 \) coordinates can be calculated as shown in 4. There are thus four
different coordinate values that can appear after the third stage. Fig. 4, see the end of the paper, shows the architecture when the three first stages are eliminated. The three stages are eliminated to the cost of six MUXes.

Continuing with the fourth stage gives the equations in 5. The number of equations in 5 is reduced but they follow the same pattern as in the second and third stage. The elimination of the fourth stage results in eight fixed values that should be crosswise distributed to the \( x_4 \) and \( y_4 \) vector coordinates. This can be solved by using 14 MUXes in three MUX-stages.

\[
\begin{align*}
\text{if } sgn_1 &= 0, sgn_2 = 0, sgn_3 = 0 \\
\begin{cases}
y_4 &= y_3 + \frac{x_3}{8} = \frac{3x + x}{2} = \frac{95x}{64} \\
x_4 &= x_{3} - \frac{y_3}{8} = \frac{-3x + x}{2} = \frac{29x}{64}
\end{cases}
\end{align*}
\]

if \( sgn_1 = 1, sgn_2 = 1, sgn_3 = 1 \)

\[
\begin{align*}
\begin{cases}
y_4 &= y_3 - \frac{x_3}{8} = \frac{x - 3x}{2} = \frac{95x}{64} \\
x_4 &= x_{3} + \frac{y_3}{8} = \frac{3x + x}{2} = \frac{29x}{64}
\end{cases}
\end{align*}
\] (5)

The removal of a fifth stage will result in 16 fixed values, which requires 30 MUXes. In that case, it can be questioned if it is not more efficient to use a ROM instead.

V. RESULTS

The gain in the complexity reduction can for instance be measured in cell area or number of transistors. Here the number of transistors is chosen. The adder/subtractor consist of an adder cell and an XOR-gate, here using 28 respectively 10 transistors each. MUXes using 10 transistors each and inverters of two transistors is used as well. The cell counts will be as shown in Table I, if a wordlength of 8 bits is assumed along the stages. The original CORDIC corresponds to the first row, i.e. there are no eliminated stages.

<table>
<thead>
<tr>
<th>No. of reduced stages</th>
<th>ADD</th>
<th>XOR</th>
<th>INV</th>
<th>MUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>18</td>
<td>18</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>16</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>14</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>12</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>10</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>8</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

From the table it is reasonable to expect that there is a complexity minimum since the adders are decreasing linearly and the MUXes are increasing in an exponential manner. Table II shows the number of transistors based on the number of cells in Table I. The reduction percentage is also shown in the table.

<table>
<thead>
<tr>
<th>No. of reduced stages</th>
<th>No. of transistors</th>
<th>Reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5482</td>
<td>100.0</td>
</tr>
<tr>
<td>1</td>
<td>4874</td>
<td>88.9</td>
</tr>
<tr>
<td>2</td>
<td>4424</td>
<td>80.7</td>
</tr>
<tr>
<td>3</td>
<td>4134</td>
<td>75.4</td>
</tr>
<tr>
<td>4</td>
<td>4164</td>
<td>76.0</td>
</tr>
<tr>
<td>5</td>
<td>4834</td>
<td>88.1</td>
</tr>
</tbody>
</table>

The table shows that there is a minimum at three eliminated stages. Eliminating four stages is almost as good as three stages.

VI. CONCLUSIONS

A complexity reduction methodology for unrolled CORDIC architectures is shown. The Methodology is based on removing stages to the cost of a number of MUXes. However, the number of MUXes grows exponentially, which leads to a minimum in complexity reduction. The paper shows that 25% of the circuitry can be removed when three stages are eliminated.

REFERENCES


