Improving the computational efficiency of lock-in algorithms through coherent averaging

Matias Javier Oliva 1, Pablo Andrés García 1, Enrique Spinelli 1, and Alejandro Luis Veiga 1

1Affiliation not available

October 31, 2023

Abstract

The lock-in amplifier is a commonly used technique for the processing of noisy signals with a known periodicity that involves multiplying the signal with one or more reference signals of the same periodicity and low-pass filtering the results. This enables the recovery of the amplitude and phase of the signal being measured. Another well studied technique to reduce the noise of periodic signals is the coherent averaging method, where the samples of a noisy signal with a known periodicity are averaged coherently. In this paper, a novel system combining both algorithms is proposed. In this approach, the signal is first averaged coherently for a number of cycles (Nca) and then passed through a conventional lock-in with a moving average filter of a whole number of periods of the signal (Nma) as a lowpass filter. In this scenario, the question of how to distribute the incoming samples between the coherent average and the conventional lock-in scheme arises. The mathematical aspects of this issue were evaluated, leading to the conclusion that the calculation results remain identical as long as the product NcaNma remains constant. However, it was observed that the number of operations required for each algorithm varies. By maximizing the number of samples used for the coherent average, the number of multiplications involved can be drastically reduced from NcaNmaM to just M, where M represents the number of samples in a single period of the signal. The drawbacks are the need for space to store the averaged signal, a slower convergence to the result, and an extra cycle in the calculations. Any lock-in system with moving average filter can take advantage of these results. Particularly in this study, they are employed to nearly double the achievable clock frequency of a lock-in system implemented on a Cyclone V FPGA, taking it from 119.39 MHz to 216.54 MHz without the use of hardware embedded multipliers.
Improving the computational efficiency of lock-in algorithms through coherent averaging

Matías Javier Oliva    Pablo Andrés García    Enrique Mario Spinelli    Alejandro Luis Veiga

Abstract—The lock-in amplifier is a commonly used technique for the processing of noisy signals with a known periodicity that involves multiplying the signal with one or more reference signals of the same periodicity and low-pass filtering the results. This enables the recovery of the amplitude and phase of the signal being measured. Another well-studied technique to reduce the noise of periodic signals is the coherent averaging method, where the samples of a noisy signal with a known periodicity are averaged coherently. In this paper, a novel system combining both algorithms is proposed. In this approach, the signal is first averaged coherently for a number of cycles \((N_{ca})\) and then passed through a conventional lock-in with a moving average filter of a whole number of periods of the signal \((N_{ma})\) as a low-pass filter. In this scenario, the question of how to distribute the incoming samples between the coherent average and the conventional lock-in scheme arises. The mathematical aspects of this issue were evaluated, leading to the conclusion that the calculation results remain identical as long as the product \(N_{ca}N_{ma}\) remains constant. However, it was observed that the number of operations required for each algorithm varies. By maximizing the number of samples used for the coherent average, the number of multiplications involved can be drastically reduced from \(N_{ca}N_{ma}M\) to just \(M\), where \(M\) represents the number of samples in a single period of the signal. The drawbacks are the need for space to store the averaged signal, a slower convergence to the result, and an extra cycle in the calculations. Any lock-in system with moving average filter can take advantage of these results. Particularly in this study, they are employed to nearly double the achievable clock frequency of a lock-in system implemented on a Cyclone V FPGA, taking it from 119.39 MHz to 216.54 MHz without the use of hardware embedded multipliers.

Index Terms—Lock-in, Coherent average, Moving Average Filter, Computational efficiency, FPGA, Digital Design.

I. INTRODUCTION

Lock-in amplifiers use coherent detection methods to perform module and phase measurements over a specific frequency in low signal-to-noise ratio (SNR) environments. To do so, they excite the system under test with the desired frequency \((\omega_r)\), and multiply the resultant signal with a sinusoidal reference at the same frequency. This multiplication produces a signal with two frequential components: one at \(2\omega_r\) and one at base band. This base band component carries the information of the signal’s module, so by low-pass filtering the multiplied signal, the module can be determined [1]. To recover the signal’s phase an additional reference signal, phase-shifted at the quadrature of the original reference, may be used [2] [3] [4]. This is known as a two phase lock-in amplification, and it is shown schematically in Fig. 1.

A key element in the lock-in amplification is the selection of the low-pass filter [5]. The purpose of this filter is twofold: firstly, it must eliminate the double frequency component in the multiplied signal. Secondly, it is in charge of reducing the noise that gets involved in the calculation. Considering the system’s noise as white, the measurement noise in lock-in detection is directly proportional to the square root of the bandwidth of the selected low-pass filter [6] [7]. The combination of multiplication and low-pass filtering behaves like a band-pass filter centered at \(\omega_r\), without the difficulties often associated with designing narrow band-pass filters [8].

An interesting option for the lock-in technique is a simple moving average filter (MAF). This filter has the property of presenting zeros at frequencies that are multiples of the signal’s frequency, as long as a whole number of periods of the signal are integrated [9]. This means that it can completely block the \(2\omega_r\) component. For this filter, the bandwidth is inversely proportional to the number of samples being averaged, so if it is used in the lock-in technique, the measurement noise can be effectively reduced by averaging a larger number of periods of the signal [10] [11].

Previous work [12] has shown that, if a signal perturbed with white noise of standard deviation \(\sigma_i\) enters a double phase lock-in with \(N\) cycles MAF, the standard deviation of the amplitude at the output \((\sigma_o)\), depends on the number of points being averaged and \((\sigma_i)\) through equation 1.

\[
\sigma_o = \frac{\sigma_i}{\sqrt{M/N}}
\]  

The coherent averaging method (CA) is an alternative technique commonly used in the processing of noisy periodic signals. In this approach, the samples of a noisy signal with a known periodicity are averaged coherently. This means that
the samples that occupy the same position relative to the total number of samples in a period are averaged together. This has the effect of effectively reducing the noise in the signal [13] [14].

The method suggested in this article is to perform coherent averaging prior to the multiplication and low-pass filtering stages of a conventional lock-in scheme with MAF. In this scenario, schematically shown in Fig. 2, with the signal passing through a coherent averaging of \( N_{ca} \) cycles stage, then a multiplication stage, and finally a moving average of \( N_{ma} \) cycles stage, the question of how to divide the incoming samples between the CA and the MAF processing arises, i.e., how to properly select \( N_{ca} \) and \( N_{ma} \).

This paper will demonstrate that performing traditional lock-in detection with MAF during \( N \) signal cycles is mathematically equivalent to performing CA during \( N_{ca} \) cycles, followed by lock-in detection with MAF during \( N_{ma} \) cycles, as long as \( N = N_{ma}N_{ca} \). After that, the number of operations between different methods will be analyzed, concluding that the most efficient way of performing the operations is with \( N_{ca} = N \) and \( N_{ma} = 1 \). Finally, these results will be used to improve the computational efficiency of a lock-in system implemented on a Cyclone V SoC-FPGA.

II. MATHEMATICAL FORMULATION

A. Core concepts

Consider the periodic signal \( s[n] \), of period \( M \), such that \( s[n] = s[n+kM] \), for any integer \( k \). This signal may be used to perform single-phase lock-in detection with MAF during \( N \) periods of the signal in an incoming arbitrary signal \( x[n] \). This operation can be described mathematically with equation 2.

\[
Y_1[n] = \frac{1}{NM} \sum_{k=n}^{n+NM-1} x[k] s[k] \tag{2}
\]

Now consider that the arbitrary signal is first averaged coherently over \( N_{ca} \) periods of the signal \( s[n] \). Then the averaged signal enters a classic lock-in detection system with a MAF of \( N_{ma} \) periods. For this last operation, \( N_{ma} \) periods of the signal are needed, where each one is the coherent average of \( N_{ca} \) periods.

The samples of the first period of coherently averaged signal \( Y_{ca}[n] \) can be expressed as shown in equation 3.

\[
Y_{ca}[n] = \frac{1}{N_{ca}} \sum_{i=0}^{N_{ca}-1} x[n+iM] \quad 0 \leq n < M \tag{3}
\]

The second period of \( Y_{ca} \) is obtained by averaging the second \( N_{ca} \) periods of the incoming signal. So if \( n \) is between \( M \) and \( 2M \), the average starts from \( (N_{ma} - 1)N_{ca} \) and goes on for \( N_{ca} \) periods. Now the relative position inside the period is given by \( l = n - M \), which is the position of the sample relative to the \( M \)-point period.

\[
Y_{ca}[n] = \frac{1}{N_{ca}} \sum_{i=0}^{N_{ca}-1} x[N_{ca}M+l+iM] \quad M \leq n < 2M \tag{4}
\]

The last period of the coherently averaged signal will be the one where \( n \) is between \( (N_{ma} - 1)M \) and \( N_{ma}M \). To obtain this signal, the averaging must start from \( (N_{ma} - 1)N_{ca}M \) and go on for \( N_{ca} \) periods. Now the relative position inside the period is given by \( l = n - (N_{ma} - 1)M \).

Then the general expression for \( Y_{ca} \) can be derived as shown in equation 5, with the indices displayed in equation 6.
Equation 8 summarizes the operation over the incoming signal \( s \) with a reference signal phase-shifted at the quadrature of \( n \). Without losing generality, since the calculations are the same, let us perform a factorization on equation 2.

The factor that is multiplied by each \( s \) is precisely the coherent average of the signal \( x[n] \) during \( N \) periods. Therefore, the expression can be rewritten as:

\[
Y[n] = \frac{1}{MN} \sum_{k=0}^{n+N-1} x[k] s[k] = \frac{1}{MN} (s[n] x[n] + \ldots + s[n+N-1] x[n+N-1] + \ldots)
\]

Taking an external summation, the previous equation yields:

\[
Y[n] = \frac{1}{MN} \sum_{k=0}^{n+N-1} \sum_{i=0}^{N-1} s[k] x[k+iM] \quad (12)
\]

Which is the equation 8 in the particular case that \( N_{ma} = 1 \) and \( N_{ca} = N \). Although this does not prove that the equations 2 and 8 are equivalent, it does give an intuitive idea of why the equality holds.

C. Formal demonstration

The formal demonstration of the equivalence between equations 2 and 8 can be derived by expanding the equation 8, and taking advantage of the periodicity of the \( s[n] \) function to perform a convenient factorization in a similar way to the one developed for the previous case. This rather tedious task will be developed in the supplemental material of this article.

B. Intuitive approach

To gain an intuitive understanding of why these expressions are equivalent, let us perform a factorization on equation 2.

\[
Y_{ca}[n] = \begin{cases} \frac{1}{N_{ca}} \sum_{i=0}^{N_{ca}-1} x[k_{0}(i,n)] & 0 \leq n < M \\ \frac{1}{N_{ca}} \sum_{i=0}^{N_{ca}-1} x[k_{1}(i,n)] & M \leq n < 2M \\ \frac{1}{N_{ca}} \sum_{i=0}^{N_{ca}-1} x[k_{2}(i,n)] & 2M \leq n < 3M \\ \vdots & \\ \frac{1}{N_{ca}} \sum_{i=0}^{N_{ca}-1} x[k_{(N_{ma}-1)}(i,n)] & (N_{ma} - 1)M \leq n < N_{ma}M \\ \end{cases}
\]

We can factorize equation 8 as follows:

\[
k_{0}(i,n) = n + iM \\
k_{1}(i,n) = N_{ca}M + (n - M) + iM \\
k_{2}(i,n) = 2N_{ca}M + (n - 2M) + iM \\
\vdots \\
k_{(N_{ma}-1)}(i,n) = (N_{ma} - 1)N_{ca}M + (n - (N_{ma} - 1)M) + iM
\]

Now this coherently averaged signal enters lock-in detection with MAF of \( N_{ma} \) periods. For this, it is first multiplied by the periodic signal \( s[n] \) and then averaged for \( N_{ma} \) periods. Equation 8 summaries the operation over the incoming signal \( x[n] \) for the combination of CA and lock-in with MAF.

\[
Y_{2}[n] = \frac{1}{MN_{ma}} \sum_{k=n}^{n+N_{ma}-1} s[k] \frac{1}{N_{ca}} \sum_{h=0}^{N_{ca}-1} \sum_{i=0}^{N_{ca}-1} x[k+M(i+h(N_{ma}-1))] \cdot (hM \leq k < (h+1)M) \quad (8)
\]

In the next sections, it will be shown that the expressions 2 and 8 (signals \( Y_{1}[n] \) and \( Y_{2}[n] \)) are mathematically equivalent as long as \( N_{ca}N_{ma} = N \).

Although the demonstration is performed with the single-phase lock-in, it can be applied to the dual-phase lock-in without losing generality, since the calculations are the same, with a reference signal phase-shifted at the quadrature of \( s[N] \).
D. Number of operations

Although, from a mathematical point of view, equations 2 and 8 are equivalent, they don’t involve the same operations. The constant term $\frac{1}{2}$, and the final calculations of amplitude and phase will be ignored for the next deductions since they don’t need to be calculated in real-time (they can be computed after the whole signal is obtained and processed).

In the case of the first method, the classic lock-in approach with MAF during $N = N_{ma}N_{ca}$, $MN$ multiplications and $MN - 1$ sums are required.

For the second method, the coherent average of $N_{ca}$ periods of the signal followed by a lock-in with MAF of $N_{ma}$ periods, each sample of the coherently averaged signal requires $N_{ca} - 1$ sums. $MN_{ma}$ samples are needed for the lock-in, so obtaining the coherently averaged signal requires $MN_{ma}(N_{ca} - 1)$ sums. Then this signal enters the lock-in, which requires $MN_{ma}$ multiplications and $MN_{ma} - 1$ sums.

In total, this sums up for $MN_{ma}$ multiplications and $MN_{ma}(N_{ca} - 1) + N_{ma}M - 1 = MN_{ma}N_{ca} - 1 = NM - 1$ sums. This means that both methods require the same number of sums, but the second one drastically reduces the number of multiplications needed, from $MN_{ma}N_{ca}$ to $MN_{ma}$.

The computational advantages of using coherent averaging prior to lock-in amplification are even clearer when a two phase lock-in is needed. In this case, the arbitrary signal is multiplied by an additional cosenoidal reference signal. If one were to conduct the calculations with a traditional lock-in with MAF, equation 2 must be calculated twice, one with each reference signal. This means $2MN$ multiplications and $2(MN - 1)$ sums.

On the other hand, the coherently averaged signal is the same for the two references, so it has to be calculated only once ($MN_{ma}(N_{ca} - 1)$ sums). Then this coherently averaged signal enters the dual-phase lock-in, which requires $2MN_{ma}$ multiplications and $2(MN_{ma} - 1)$ sums, for a total of $MN_{ma}(N_{ca} - 1) + 2(MN_{ma} - 1) = MN_{ma}(N_{ca} + 1) - 2$ sums, almost half the sums when $N_{ca}$ is big enough. These results are summarized in table I.

From this analysis, it results that to keep the number of multiplications at a minimum, $N_{ma}$ must be as small as possible. The most convenient case is to leave all the averaging to the CA system, setting $N_{ca} = N$ and $N_{ma} = 1$. This is, in fact, the case of the factorization carried out in section II-B. The drawbacks of this method are that the CA signal needs memory to be stored, that the convergence of the solution is slower; one has to necessarily wait until the end of the processing to get the results, while in the classic lock-in approach one can take intermediate results that approximate the required results, and that an extra cycle in the calculations is required. This is caused by the $M$ clock cycles needed to perform the multiplications and MAF once the whole signal has been acquired and coherently averaged.

These conclusions can be applied to any lock-in system, as long as it uses moving average as low-pass filter. In the upcoming sections, they will be utilized to enhance a system based on a field-programmable gate array (FPGA).

Going forward in this paper, the lock-in system with $N$ cycles MAF will be referred to as the LI system, and the coherent average of $N$ cycles of the signal, followed by a one cycle lock-in with MAF will be referred to as the CA-LI system.

III. Implementation on FPGA

A. Classic lock-in approach

The LI system may be implemented in logic following a block diagram like the one shown in Fig. 3.

The signal of interest, quantized in $Q_x$ bits, enters the lock-in, where it is multiplied by the reference signal, quantized in $Q_{ref}$ bits. This produces an output of $Q_1$ bits, which in the worst-case scenario will be equal to $Q_x + Q_{ref}$. Then the multiplied signal enters the MAF stage, where it is summed with up to $NM$ multiplied samples. In the worst-case scenario, each sample will be equal to $2^{Q_1}$, so the sum of M $N$ of these samples will produce an output of $Q_2 = Q_1 + \log_2(NM)$ bits. As an example, let us consider that the $Q_x = 14$ and $Q_{ref} = 16$. The implementation of a classic lock-in system requires:

- A multiplier with a $Q_1 = Q_x + Q_{ref} = 30$ bit output.
- The time to compute this operation will be defined as $T_{mult_1}$.

![Fig. 3. Block diagram for lock-in amplifier with moving average filter.](image-url)
- An adder with $Q_2 = Q_1 + \log_2(NM) = 30 + \log_2(N) + \log_2(M)$ bit output. The time to compute this operation will be defined as $T_{add1}$.
- Memory to save the reference: M samples of $Q_{ref} = 16$ bits.

If these operations are executed sequentially for each incoming sample (without the registers indicated on Fig. 3) the system won’t be ready to receive a new sample until the multiplication and addition are finished, which will take approximately $T_{mult1} + T_{add1}$, plus the time to access the reference from memory.

On the other hand, by placing intermediate registers, three independent and consecutive tasks can be defined:

- Register incoming sample ($X_{[i]}$) and corresponding reference ($R_{[i]}$).
- Register product ($P_i = X_{[i]}R_{[i]}$).
- Register sums ($Y = Y + P_i$).

These operations can be implemented in a pipeline, as shown in Table II, maximizing the achievable clock frequency of the system. Additionally, these intermediate registers allow the FPGA compiler to implement the multiplications in a more efficient manner, taking advantage of DSP blocks, i.e., hardware specifically intended for these types of operations [15]. The downside of this approach is a 3 clock cycle delay on the output, which is acceptable in the majority of cases.

In this pipeline, the clock frequency can go as fast as the slowest of the tasks, which is reasonable to assume is the $T_{mult1}$, so the maximum achievable clock frequency will be $\frac{1}{T_{mult1}}$. This assumption will be revisited and demonstrated in the next sections of this article.

### B. Coherent average followed by lock-in

The CA_LI system (coherent average of $N_{ca}$ cycles followed by 1 cycle lock-in) can be implemented in logic following a block diagram like the one shown in Fig. 4. This block diagram can be subdivided into two stages: the CA stage and the lock-in stage. The interesting part of this segmentation is that both systems can be executed with different clocks.

The signal of interest, quantized in $Q_x$ bits, enters the CA system, where it is added with the last $N_{ca}$ samples that occupy the same relative position in the period. In the worst-case scenario, $Q_3 = Q_x + \log_2(N_{ca})$ bits will be necessary to store each sample of the coherently averaged signal. Once the $N_{ca}$ periods of the signal are averaged, the signal enters a classic lock-in, like the one described in section III-A. To implement this system on hardware, one needs:

- To implement the coherent average of $N_{ca}$ periods system (operating at clock 1):
  - Logic to select the sample that must be added to the incoming sample.
  - An adder of $Q_3 = Q_x + \log_2(N_{ca}) = 30 + \log_2(N_{ca})$ bits.
  - Memory to store the coherently averaged signal: M samples of $Q_3$ bits.

- To implement the lock-in with MAF of one period (operating at clock 2):
  - A multiplier with a $Q_4 = Q_x + Q_{ref} + \log_2(N_{ca}) = 30 + \log_2(N_{ca})$ bit output.
  - An adder with $Q_5 = Q_x + Q_{ref} + \log_2(N_{ca}) + \log_2(M) = 30 + \log_2(N_{ca}) + \log_2(M)$ bit output.
  - Memory to save the reference: M samples of $Q_{ref}$ bits.

For the first stage of the algorithm, three independent tasks may be defined by placing intermediate registers and a memory, as indicated in Fig. 4. These are:

- Fetch: Register incoming sample ($X_{[i]}$) and the corresponding averaged sample from the memory ($Y_{ca[i]}$). The time to perform this operation will be referred as $T_{fetch}$.
- Add: Register the sum of the incoming sample and the corresponding reference ($R_{[i]}$). The time to perform this operation will be referred as $T_{save}$.
- Save: Save the new averaged sample in memory ($Y_{ca} = S$). The time to perform this operation will be referred as $T_{add2}$.

Similarly to the previous case, these operations can be implemented in a three stage pipeline, as shown in table III. It is reasonable to assume that the sum operation is the slowest of these tasks, so the maximum achievable frequency for clock 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reg. sample ($T_{reg}$)</td>
<td>0</td>
</tr>
<tr>
<td>$X_{[0]}$</td>
<td>$X_{[1]}$</td>
</tr>
<tr>
<td>Reg. product ($T_{mult1}$)</td>
<td>-</td>
</tr>
<tr>
<td>Reg. accumulator ($T_{add1}$)</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table II</th>
<th>LOCK-IN PIPELINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>Clock</td>
</tr>
<tr>
<td>Reg. sample ($T_{reg}$)</td>
<td>0</td>
</tr>
<tr>
<td>$X_{[0]}$</td>
<td>$X_{[1]}$</td>
</tr>
<tr>
<td>Reg. product ($T_{mult1}$)</td>
<td>-</td>
</tr>
<tr>
<td>Reg. accumulator ($T_{add1}$)</td>
<td>-</td>
</tr>
</tbody>
</table>

Fig. 4. Block diagram for CA followed by LIA with MAF.
LI algorithms can be used to estimate the CA

Expected

\[ \sigma_i = 0.2887 \]

with conventional lock-in (LI) and coherent average followed by lock-in (CA_LI), with \( N = 4 \). made.

Let us consider a sinusoidal signal of amplitude \( A = 1 \) and \( M = 128 \) points per cycle, perturbed with uniform noise of \( \sigma_i \) standard deviation as the input of the system. For the reference, let us use sine and cosine waves, calculated with \( M \) points per cycle and \( A_{ref} \) amplitude. Under these conditions, the dual-phase LI and CA_LI algorithms can be used to estimate the amplitude of the underlying signal. Depending on the election of \( N \), this estimation will be more or less accurate, but the estimation with both methods will always be the same. To emphasize this, let’s examine Fig. 5 and Table IV.

Fig. 5 shows the estimation of the amplitude of the signal using both algorithms as the time passes and the signal enters the algorithm, one sample at each clock tick. The final estimation is the same, as expected, and it is very close to the expected \( A = 1 \), but the intermediate calculus are quite different. The LI converges to the result more gradually and provides the results one cycle earlier. The downside of this approach is the required number of multiplications, as noted in previous sections. The CA_LI on the other hand, takes one extra cycle and converges abruptly to the results.

Table IV shows the outputs of the algorithms for different values of \( N \) at a fixed signal-to-noise ratio. For the estimation of the amplitude, 100 realizations of the calculations were

### TABLE IV
**Simulation results**

<table>
<thead>
<tr>
<th>( N )</th>
<th>( A_{LI} )</th>
<th>LI ( a^n ) of products</th>
<th>( A_{CA,LI} )</th>
<th>CA_LI ( a^n ) of products</th>
<th>( \sigma_o )</th>
<th>Expected ( \sigma_o )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.0183</td>
<td>512</td>
<td>1.0183</td>
<td>256</td>
<td>0.21474</td>
<td>0.25516</td>
</tr>
<tr>
<td>4</td>
<td>0.97311</td>
<td>1024</td>
<td>0.97311</td>
<td>256</td>
<td>0.16137</td>
<td>0.18042</td>
</tr>
<tr>
<td>8</td>
<td>1.0012</td>
<td>2048</td>
<td>1.0012</td>
<td>256</td>
<td>0.12147</td>
<td>0.12758</td>
</tr>
<tr>
<td>16</td>
<td>0.99393</td>
<td>4096</td>
<td>0.99393</td>
<td>256</td>
<td>0.090211</td>
<td>0.090211</td>
</tr>
<tr>
<td>32</td>
<td>1.0009</td>
<td>8192</td>
<td>1.0009</td>
<td>256</td>
<td>0.062989</td>
<td>0.062989</td>
</tr>
<tr>
<td>64</td>
<td>1.002</td>
<td>16384</td>
<td>1.002</td>
<td>256</td>
<td>0.040405</td>
<td>0.040405</td>
</tr>
<tr>
<td>128</td>
<td>0.99883</td>
<td>32768</td>
<td>0.99883</td>
<td>256</td>
<td>0.031676</td>
<td>0.031676</td>
</tr>
<tr>
<td>256</td>
<td>1.0004</td>
<td>65536</td>
<td>1.0004</td>
<td>256</td>
<td>0.021749</td>
<td>0.022553</td>
</tr>
<tr>
<td>512</td>
<td>0.99962</td>
<td>131072</td>
<td>0.99962</td>
<td>256</td>
<td>0.016181</td>
<td>0.015947</td>
</tr>
<tr>
<td>1024</td>
<td>0.99945</td>
<td>262144</td>
<td>0.99945</td>
<td>256</td>
<td>0.011478</td>
<td>0.011276</td>
</tr>
</tbody>
</table>

*For all cases \( A = 1 \), \( M = 128 \) and \( \sigma_i = 2.8868 \)

**\( \sigma_o \) and the amplitude are calculated considering 100 realizations

*** Expected \( \sigma_o \) is calculated using expression 1
made, and the mean value of the amplitude estimations was taken and labeled as $A_{LI}$ and $A_{CA, LI}$ for each algorithm. The standard deviation of this amplitude was also taken and labeled as $\sigma_o$. In all cases, both LI and CA_LI deliver the same results, and the values of $\sigma_o$ behave as expected from equation 1.

B. Individual timing per operation

In order to quantify the assumptions made in Section III about the timing of each operation, some simple systems were implemented and tested on a Cyclone V FPGA.

- For LI system:
  - Multiplier with $Q_x + Q_{ref} = 30$ bit output ($T_{mult_1}$).
  - Adder with $Q_x + Q_{ref} + \log2(NA)$ bit output ($T_{add_1}$).

- For the CA_LI system:
  - Adder with $Q_x + \log2(N_{ca})$ ($T_{add_2}$).

The operations of the one cycle lock-in in CA_LI system (those corresponding to $T_{mult_1}$ and $T_{add_1}$) don’t affect the algorithms performance, as stated earlier; hence, the timing of these modules was not tested.

The parameters M and N are application-dependent, as their values vary based on the specific requirements and characteristics of the application. For this timing estimation, let us assume that $N \leq 8192$ and $M \leq 128$. The values of $Q_{ref}$ and $Q_x$ used are the ones proposed in Section III, 16 and 14 bits, respectively. With these values, $T_{add_1}$ becomes an addition with a 50-bit output, and $T_{add_2}$ an addition with a 27-bit output.

For each operation, the maximum achievable frequency was calculated using the tools provided by Intel’s Quartus II software. For the multiplications, two different options are available: implementing them on DSP blocks or implementing them with logic elements. Both scenarios were considered since the DSP block allow for better performance but is a limited resource. Table V summarizes the results.

This analysis confirms some of the assumptions made in Section III. The bottleneck of the LI system is indeed the multiplication (141.3 MHz), as long as DSP blocks are not used. However, if they are used, the 50-bit addition limits the operation of the system (215.24 MHz). The CA_LI system, on the other hand, has its performance limited by access to memory, in particular the time to fetch a sample from memory (279.25 MHz). In both cases (with or without DSP blocks), this approach is more efficient than the traditional lock-in with MAF.

It is important to highlight that these timing values were calculated under ideal conditions, with only one operation at a time in the design. However, the actual timing details of the implementation of a complete lock-in system with each approach will approximate these theoretical values, as shown in subsequent sections.

C. Lock-in implementation

The LI and CA_LI systems were implemented in the Cyclone V FPGA, following the design guidelines provided in Section III. For each algorithm, the maximum frequency achievable was calculated with and without the use of DSP blocks, as shown in Table VI.

These results are in line with the assumptions made in previous sections. It is important to notice that even though the $clk_2$ of the CA_LI system is the most restrictive one, it is not working with the signal in real time, so the CA_LI system outperforms the LI system even without the need for any DSP blocks.

The resource utilization of each algorithm under different circumstances is shown in Table. VII. Based on this table, it

---

**TABLE V**

<table>
<thead>
<tr>
<th>Operation</th>
<th>DSPs</th>
<th>Max Freq [MHz]</th>
<th>Related Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplication [30 bit]</td>
<td>✓</td>
<td>310.08</td>
<td>$T_{mult_1}$</td>
</tr>
<tr>
<td>Multiplication [30 bit]</td>
<td>☐</td>
<td>141.3</td>
<td>$T_{mult_2}$</td>
</tr>
<tr>
<td>Addition [50 bit]</td>
<td>☐</td>
<td>215.24</td>
<td>$T_{add_2}$</td>
</tr>
<tr>
<td>Addition [27 bit]</td>
<td>☑</td>
<td>320.92</td>
<td>$T_{add_3}$</td>
</tr>
<tr>
<td>Fetch Sample</td>
<td>☑</td>
<td>315.06</td>
<td>$T_{fetch}$</td>
</tr>
</tbody>
</table>

*Same result with or without DSP blocks

**TABLE VI**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>DSPs</th>
<th>Max Freq [MHz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LI</td>
<td>✓</td>
<td>159.21</td>
</tr>
<tr>
<td>LI</td>
<td>✓</td>
<td>119.39</td>
</tr>
<tr>
<td>CA_LI [clk1]</td>
<td>X</td>
<td>216.54</td>
</tr>
<tr>
<td>CA_LI [clk2]</td>
<td>✓</td>
<td>143.33</td>
</tr>
<tr>
<td>CA_LI [clk3]</td>
<td>X</td>
<td>90.32</td>
</tr>
</tbody>
</table>

*Same result with or without DSP blocks

**TABLE VII**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>DSPs</th>
<th>M</th>
<th>ALMS</th>
<th>ALUTs</th>
<th>Registers</th>
<th>Memory bits</th>
<th>M10K</th>
<th>DSP blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>LI</td>
<td>✓</td>
<td>32</td>
<td>111.8</td>
<td>222</td>
<td>360</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>LI</td>
<td>X</td>
<td>32</td>
<td>357.4</td>
<td>680</td>
<td>259</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LI</td>
<td>✓</td>
<td>128</td>
<td>137.2</td>
<td>268</td>
<td>336</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>LI</td>
<td>X</td>
<td>128</td>
<td>382.9</td>
<td>726</td>
<td>580</td>
<td>1728</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>CA_LI</td>
<td>✓</td>
<td>32</td>
<td>229.7</td>
<td>387</td>
<td>586</td>
<td>6912</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>CA_LI</td>
<td>X</td>
<td>32</td>
<td>940.2</td>
<td>1604</td>
<td>686</td>
<td>1728</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>CA_LI</td>
<td>✓</td>
<td>128</td>
<td>263.9</td>
<td>437</td>
<td>586</td>
<td>6912</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>CA_LI</td>
<td>X</td>
<td>128</td>
<td>963.5</td>
<td>1654</td>
<td>665</td>
<td>6912</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Values obtained with N=8192
is clear that the CA_LI algorithm is more resource-intensive, particularly in terms of memory utilization, since it needs space to save the coherently averaged signal. However, it is more cost-effective in terms of DSP block utilization. In this matter, it is worth emphasizing that the DSP blocks used for the CA_LI system are employed on the one-cycle lock-in, which has no impact on the performance of the system, so they could be safely eliminated.

Both the Verilog code and the simulation code are open source, licensed under the terms of the MIT license, and available at [16].

V. Conclusions

It has been shown that the combination of the coherent average of $N_{ca}$ cycles of a signal of interest and a conventional lock-in system with moving average filter of $N_{ma}$ cycles (CA_LI system) shields the same results as a lock-in of $N$ cycles of the signal (LI system), as long as $N = N_{ca}N_{ma}$.

Although both methods give the same results, the computations required to reach them are quite different. Specifically, the number of multiplications needed is reduced as $N_{ca}$ becomes larger (and $N_{ma}$ smaller). The most efficient case is obtained when $N_{ca} = N$ and $N_{ma} = 1$, where the required number of multiplications gets fixed to M, regardless of the selected N.

These insights can be applied to any lock-in system. In particular, this article has shown how they can be applied to improve the computational efficiency of a lock-in system implemented on an FPGA. The application of the coherent average algorithm allows to move the bottleneck of the implementation from a multiplication to a simple addition, improving the maximum achievable frequency of the system.

In a test performed on a Cyclone V device, the frequency improvement achieved was from 119.39 MHz to 216.54 MHz without the use of embedded hardware to perform the multiplications (DSP blocks) and from 159.21 MHz to 216.54 MHz if DSP blocks are used. These are very interesting findings considering that DSP blocks are a quite limited resource. The downsides are a larger embedded memory utilization since the averaged data needs to be stored to perform the computations, a slower convergence to the results, and an extra cycle of calculations.

These conclusions open the path to the implementation of very high-frequency, multi-channel and multi-frequency coherent detection methods on mid-range devices, providing cost-effective solutions to many signal processing applications.

References


SUPPLEMENTAL MATERIAL

To prove the equivalence between equations 13 and 14 (equations 2 and 8 in the article) let us expand the expression 14.

\[ Y_1[n] = \frac{1}{N M} \sum_{k=n}^{n+N M-1} x[k] s[k] \quad (13) \]

\[ Y[n] = \frac{1}{M N_{ca}} \sum_{k=n}^{n+N_{ma} M-1} \frac{1}{N_{ca}} \sum_{h=0}^{N_{ma}-1} x[k+M(h(N_{ca}-1))] \cdot (h M \leq k < (h+1) M) \quad (14) \]

\[ Y[n] M N_{ca} N_{ma} = \sum_{k=n}^{n+N_{ma} M-1} s[k] \sum_{h=0}^{N_{ma}-1} (x[k+M(h(N_{ca}-1))] + x[k+M(1+h(N_{ca}-1))] + \ldots + x[k+M((h(N_{ca}-1)+h(N_{ca}-1))] \cdot (h M \leq k < (h+1) M) \]

\[ Y[n] M N_{ca} N_{ma} = \sum_{k=n}^{n+N_{ma} M-1} s[k] \sum_{h=0}^{N_{ma}-1} (x[k] + x[k+2 M] + \ldots + x[k+(N_{ca}-1) M]) \cdot (0 \leq k < M) + \ldots + \]

\[ + (x[k+(N_{ca}-1) M] + x[k+N_{ca} M] + \ldots + x[k+2(N_{ca}-1) M]) \cdot (M \leq k < 2 M) + \ldots \]

\[ \vdots \]

\[ + (x[k+M((N_{ma}-1)(N_{ca}-1))] + \ldots + x[k+M((N_{ma}-1)(N_{ma}-1))] \cdot ((N_{ma}-1) M \leq k < N_{ma} M) \]

\[ Y[n] M N_{ca} N_{ma} = s[n] x[n] + s[n+M] x[n+M] + \ldots + s[n+(N_{ca}-1) M] x[n+(N_{ca}-1) M] + s[n+1] x[n+1] + s[n+1+M] x[n+1+M] + \ldots + s[n+1+(N_{ca}-1) M] x[n+1+(N_{ca}-1) M] + \ldots \]

\[ \vdots \]

\[ + s[n+N_{ma} M-1] x[n+N_{ma} M-1+M(N_{ma}-1)(N_{ca}-1)] + \ldots + s[n+N_{ma} M-1+M((N_{ma}-1)(N_{ma}-1))] x[n+N_{ma} M-1+M((N_{ma}-1)(N_{ma}-1))] \]

Now let us take advantage of the periodicity of the \( s[n] \). Since \( s[n] = s[n+M] \) then \( s[n] x[n+M] = s[n+M] x[n+M] \).

\[ Y[n] M N_{ca} N_{ma} = s[n] x[n] + s[n+M] x[n+M] + \ldots + s[n+(N_{ca}-1) M] x[n+(N_{ca}-1) M] + \]

\[ + s[n+1] x[n+1] + s[n+1+M] x[n+1+M] + \ldots + s[n+1+(N_{ca}-1) M] x[n+1+(N_{ca}-1) M] + \]

\[ \vdots \]

\[ + s[n+N_{ma} M-1+M(N_{ma}-1)(N_{ca}-1)] x[n+N_{ma} M-1+M(N_{ma}-1)(N_{ca}-1)] + \ldots + \]

\[ + s[n+N_{ma} M-1+M((N_{ma}-1)(N_{ma}-1))] x[n+N_{ma} M-1+M((N_{ma}-1)(N_{ma}-1))] \] (15)

In each term of this last summation, the index of \( s[n] \) is always equal to the index of \( x[n] \). None of the indexes are repeated, and all of them are summed, from \( n \) to \( k_{end} = n + N_{ma} M - 1 + M((N_{ca} - 1) + (N_{ma} - 1)(N_{ca} - 1)) \). This \( k_{end} \) may be expressed as follows:

\[ k_{end} = n + N_{ma} M - 1 + M((N_{ca} - 1) + (N_{ma} - 1)(N_{ca} - 1)) = \]

\[ = n + N_{ma} M - 1 + M N_{ca} - M + M N_{ma} N_{ca} - M N_{ma} - M N_{ca} + M \]

\[ = n - 1 + M N_{ma} N_{ca} \]

So equation 15 can be rewritten as:

\[ Y(n) = \frac{1}{M N_{ca} N_{ma}} \sum_{k=n}^{k=n+M N_{ma} N_{ca}-1} s[k] x[k] \]

Which is equation 2 in the special case where \( N = N_{ma} N_{ca} \), completing the proof.