A Delay Locked Loop for Instantaneous Time-Of-Flight Sensing based on a CMOS Demodulation Detector

Robin Deleener, Hans Ingelberts, Maarten Kuijk
Laboratory for Micro- & Photonic Circuits (LAMI)
Dept. of Electronics and Informatics (ETRO), Vrije Universiteit Brussel (VUB)
Brussels, Belgium
rdeleene@etro.vub.ac.be

Abstract—A Delay-Locked Loop (DLL) is presented that can track the phase of a continuous-wave modulated light signal. Using it in a time-of-flight set-up allows tracking of an object’s distance in a fast and reliable way. The CMOS detector is built around a current-assisted photonic demodulator (CAPD) with an additional common-mode feedback (CMFB) avoiding detector saturation. The DLL control loop is implemented in an external FPGA, enabling flexibility and tuning of the control parameters. The system is a good candidate for industrial time-of-flight applications, where reaction time is important and outliers are unwanted. This novel DLL implementation has a phase noise below 1° and a settling time of 1 ms at a modulation frequency of 20 MHz.

Keywords—Current-Assisted Photonic Demodulator, Delay Locked Loops, Photonic Demodulation, Optical Detectors, Time-Of-Flight Imaging

I. INTRODUCTION

Time-of-flight (TOF), amongst others [1], is a well-known and highly used principle to create 3D-images of a scene. Several optical detectors, implementing this TOF principle have been described [2-7]. The classic approach to extract phase information from the received modulated light in these CMOS time-of-flight imagers is to perform four measurements with different modulation reference phases (0°, 90°, 180° and 270°) [8]. Each sub-measurement takes some integration time and only after all four sub-measurements are done, a distance can be calculated. During this whole measurement period, the effective distance may not change. If it does change, the distance calculation algorithm can produce false outcomes that can differ dramatically from the correct distance.

For the purpose of this paper, we selected the Current-Assisted Photonic Demodulator (CAPD) [7], as demonstration device because it has a high demodulation contrast that is created by an electric drift field in the detecting substrate. An analog common-mode feedback (CMFB) was added to keep the common mode of the differential output at a predetermined voltage avoiding early saturation [9].

This work was supported in part by the HERCULES project VeRONICA.

Fig. 1. CMFB CAPD differential output voltages for different light to demodulation phase shifts when (a) the demodulation signal is leading, (b) the demodulation signal is lagging, and (c) the loop is in lock at 90° phase difference.

II. DLL DESIGN

The DLL’s function is to keep the differential output voltage of the detector equal to zero by controlling the phase of the demodulation signal. In lock, the demodulation signal has a 90° phase shift compared to the modulation signal. The DLL’s phase shift, applied to lock both signals, is an immediate measure for the distance between the object and the detector.

The DLL is implemented on an FPGA to enable flexibility and tuning of the control parameters. This FPGA, embedded on the Altium NanoBoard NB3000 development board, is also
used to generate and synchronize control signals, to manage communication and to provide a basic user interface.

The user interface is used to read out the phase/distance and to change the settings of the DLL.

A schematic overview of the DLL (Fig. 2) shows a phase controller which is configured to read detector data from a differential ADC. The phase controller will adjust the phase of the demodulation signal with phase shifter $\Delta \Phi_A$ in an attempt to keep the ADC output at zero.

A second phase shifter $\Delta \Phi_B$ (configured by the user interface) is used for simulating object’s distance changes by shifting the laser modulation phase.

III. DLL IMPLEMENTATION

The (de-)modulation signal is frequency-programmable by using a 200 MHz reference clock with a configurable 50% duty-cycle clock divider. The modulation signal can be shifted with steps of 5 ns by a configurable delay element ($\Delta \Phi_B$), corresponding to a distance chance of 150 cm. The demodulation signal can be shifted ($\Delta \Phi_A$) with much smaller steps (100 ps) using a PLL available in the FPGA, allowing a theoretical distance resolution of 3 cm. For higher precision, smaller steps would be needed, calling for an additional adjustable delay-chip, or the use of a higher speed FPGA.

The optical detector generates a continuous differential voltage which is sampled with a 50 kSPS differential ADC. The ADC output value, refreshed at a frequency of 50 kHz, is used by the phase controller to adapt the demodulation phase $\Delta \Phi_A$ to keep the ADC output at zero.

The phase controller outputs the configured $\Delta \Phi_A$ to the user interface to calculate the distance (3). This can be derived as follows:

$$d = \frac{RTT}{2} c$$  \hspace{1cm} (1)

$$d = \frac{1}{2} \frac{(\Delta \Phi_A - 90^\circ)}{360^\circ} T_{\text{mod}} c$$  \hspace{1cm} (2)

$$d = \frac{1}{2} \frac{(\Delta \Phi_A - 90^\circ)}{360^\circ} f_{\text{mod}} c$$  \hspace{1cm} (3)

where RTT is the round trip time of the laser pulse, $c$ is the speed of light, $T_{\text{mod}}$ and $f_{\text{mod}}$ are respectively the modulation period and frequency.

The DLL’s control loop contains the phase controller that is implemented as a PI controller and a PLL (Fig. 3). Its speed of is limited by the maximum sample rate of the ADC, being 50 kHz.

The PLL is used to control the phase difference between the modulation and demodulation pulse. This phase can be shifted up or down with 100 ps increments by giving a pulse to the PLL’s up or down input. The PI controller corrects, depending on the output of the ADC, the wanted phase $\phi_w$. Knowing the current phase of the PLL $\phi_c$, the PI controller gives as many up or down pulses to set the new phase.

$$\phi_w = K_p \Delta \phi + K_i (\phi_c + \Delta \phi)$$  \hspace{1cm} (4)

$$\#\text{pulses} = \phi_w - \phi_c$$  \hspace{1cm} (5)

where $K_p$ and $K_i$ are the PI controller gain factors and $\Delta \phi$ is a measure for the phase error captured from the ADC output.

The PI controller outputs the phase (to the user interface) as the number of pulses given to the PLL to get in lock. The pulse range (i.e. the number of pulses to shift 360°) is calculated with (6).

$$\#\text{pulses}_{\text{max}} = \frac{T_{\text{mod}}}{\text{phasestep resolution}} = \frac{T_{\text{mod}}}{100 \text{ ps}}$$  \hspace{1cm} (6)

For a modulation frequency of 1 MHz, the maximum number of pulses equals 10000, which can be represented by a 14-bit value. The phase output by the DLL is therefore chosen to be a 16-bit value. It can be converted to a phase in degrees with (7).

$$\phi(^\circ) = \phi(\#\text{pulses}) \times \text{phasestep resolution} \times \frac{T_{\text{mod}}}{360^\circ}$$  \hspace{1cm} (7)

where $\phi(\#\text{pulses})$ is the 16-bit DLL value, phasestep resolution the PLL time step (100 ps) and $T_{\text{mod}}$ the modulation period.
IV. TESTING AND RESULTS

The DLL shows acceptable results for (de)modulation frequencies up to 20 MHz. For higher frequencies the phase shift of 100 ps becomes too large compared to the modulation period and the DLL loses performance. To increase the maximum modulation frequency, timing hardware with smaller time steps are needed. Increasing the modulation frequency improves amongst others, the phase oscillation and settling time but the distance range becomes smaller. A compromise should be made.

Due to the DLL’s finite reaction speed and the DLL’s phase controller implementation, the DLL will not be able to keep the differential output (“OUT+/-”) constantly at zero but will oscillate around it (Fig. 4). Due to this output voltage oscillations, the DLL’s phase will also not be constant, but will oscillate slightly around a mean value. The phase oscillation amplitude and settling time should be kept as small as possible by well choosing the DLL control parameters.

To visualize the phase oscillation, an available 8-bit DAC is used to convert the eight least significant bits of the 16-bit phase into a voltage, which can be observe together with the detector outputs by an oscilloscope. Once the DLL is settled, the upper eight bits of the phase are constant. Only during startup, the upper eight bits can change as a result of an over or underflow of the 8 least significant bits (Fig. 4(a) between time 50 and 75 ms).

The settling time when the DLL is enabled or when an abrupt phase shift occurs, depends on the modulation frequency and the gain factors \( K_p \) and \( K_i \) of the PI controller. The oscillation amplitude, \( \Phi_{osc} \), of the phase signal in TABLE I. is calculated with (8).

\[
\Phi_{osc} = \frac{\Delta V_{osc}}{V_{max,DAC}} \times 2^8 \times 100 \text{ ps} \times \frac{360^\circ}{T_{mod}}
\]  
(8)

where \( \Delta V_{osc} \) is the amplitude of the phase oscillation outputted by the DAC, \( 2^8 \) = the DAC’s resolution, 100 ps = the phasestep resolution and \( T_{mod} \) the modulation period.

The settling times are calculated as the time between the rising edge of “Trigger” (indicating the modulation phase changed or the DLL was enabled) and the time where the “Phase”-signal oscillation stays below \( \Phi_{osc} \), the steady state oscillation. Even during settling time, outliers in the phase signal were absent.

<table>
<thead>
<tr>
<th>Modulation frequency</th>
<th>( \Phi_{osc} )</th>
<th>Settling time for 10° phase shift</th>
<th>Settling time for 90° phase shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MHz</td>
<td>15°</td>
<td>10 ms</td>
<td>40 ms</td>
</tr>
<tr>
<td>5 MHz</td>
<td>8°</td>
<td>7 ms</td>
<td>10 ms</td>
</tr>
<tr>
<td>10 MHz</td>
<td>6°</td>
<td>4 ms</td>
<td>6 ms</td>
</tr>
<tr>
<td>20 MHz</td>
<td>3°</td>
<td>1 ms</td>
<td>3 ms</td>
</tr>
</tbody>
</table>

The phase oscillation amplitude can be further decreased by averaging the phase over several measurements. The mean value is calculated by taking the mean value of a FIFO buffer to keep an instantaneous measurement of the phase. By changing the FIFO buffer length, the averaging effect can be increased or decreased. A large FIFO will average more and as such decrease the oscillation amplitude, but scene changes will be observed slower, while a small FIFO will quickly detect changes but will have a larger oscillation.

With a FIFO of length 512, a phase oscillation below 1° can be achieved (Fig. 5).

![Fig. 4. Detector output voltages and DLL’s corresponding phase output with (a) the DLL enabled and (b) an abrupt phase change of 10° at the rising edge of ‘Trigger’.](image1)

![Fig. 5. DLL’s phase output oscillation for different FIFO lengths.](image2)
To conclude, a sweep of $\Delta \Phi_B$ shows that the DLL’s phase $\Delta \Phi_A$ follows the real phase over a complete modulation period (Fig. 6). This measurement is made at a modulation frequency of 20 MHz and a FIFO length of 512.

V. FUTURE WORK

To increase the performance of the DLL we will need to improve the DLL controller, i.e. the PI control loop. By tuning the control parameters and improving the DLL’s implementation we can improve the phase oscillation and settling times. Furthermore, we can add specific high-resolution timing hardware to increase the phase step resolution and modulation frequency. Another possibility is to implement an analog, on-chip DLL next to the detector.

VI. CONCLUSION

A novel DLL and its implementation were presented. Its phase output showed good correspondence to the actual phase without outliers. The current DLL implementation reaches a phase oscillation below 1°, and a settling time of 1 ms at a modulation frequency of 20 MHz. The phase oscillation can be reduced further by the use of a FIFO buffer to average the DLL’s output. More specific hardware and improved implementations should allow us to improve the DLL performance making it a good candidate for industrial time-of-flight applications, where reaction time is important and outliers are unacceptable.

REFERENCES