When driving high-speed data into a digital to analog converter (DAC) valid clock-to-data timing must be maintained regardless of variations in process, voltage and temperature (PVT). This demands that board-to-board (process/device) and operating (temperature) variations are accommodated by a single hardware and firmware configuration. Achieving this becomes increasingly difficult above modest data rates, such as 500MSa/s. Voltage variations are less dominant assuming accurately regulated supplies are used.

This application note describes a solution for implementing data rates up to the 1.3GSa/s capability of the MB86065, through a single LVDS port driven by a Xilinx® Virtex™-5 FPGA. The same concept can also be applied to the dual MB86064, up to its guaranteed minimum rate of 1GSa/s.

Implementation was demonstrated using the Fujitsu development kit, DK86065-2, and the Xilinx ML550 Networking Interface Platform. To simplify the interconnection of these two boards Fujitsu developed a passive interface adapter module.

Trademarks and registered trademarks are the property of their respective owners.
1 Overview

The problem of maintaining clock-to-data timing in a DAC application originates from the need to make the data converter ‘clock master’, where a synchronising clock is fed back to the data source. This is very different to an analog to digital converter (ADC) application where clock and data are source-synchronous and relative timing will track assuming good circuit and printed circuit board design.

Difficulties emerge in DAC applications when potential timing variations exceed the available data ‘eye’, which is almost certainly the case above 500MSa/s where the data period is only 2ns. Higher conversion rates, such as the 1.3GSa/s [~770ps data period] supported by the market leading MB86065, call for a superior solution to ensure mass produced systems do not need calibration during manufacture nor operation.

Whilst it might be possible to implement a semi-automated solution in the DAC, whereby successive data samples are buffered and scanned to identify a valid sample, such approaches suffer from introducing unknown delay variations in multiple channel designs which is undesirable in applications implementing diversity or phased-arrays. Also, such a system would not adjust itself on every sample and so this raises the question of having to decide when to resynchronise be it time or environment based, for example a step-change in operating temperature.

The Fujitsu solution has been to implement a dynamic Loop-Clock which takes advantage of the Phase Locked Loops (PLLs) and Delay Locked Loops (DLLs) incorporated in FPGAs. This functionality was originally intended to support off-chip synchronisation with high-speed memory over a double-data rate (DDR) interface and relies on the DLL or PLL being able to receive an off-chip feedback clock. In effect, the Loop-Clock architecture this makes the DAC look like a high-speed memory into which data is written but never read back.
The basis of the Loop-Clock implementation is illustrated in Figure 1. Instead of using the reference clock received from the DAC to output the data samples directly, it is fed to the reference input of an appropriate PLL or DLL. This then generates the clock which outputs the FPGA data. In addition, a loop-clock is generated which is also output as differential LVDS and fed to the DAC. The loop-clock is routed through a passive, selectable delay equivalent to that incorporated in the reference clock path inside the DAC before being routed back to the FPGA. The loop is completed by connecting this return clock to the PLL/DLL’s feedback clock input.

With the loop-clock complete, two programmable delays provide a means to adjust the relative clock-to-data timing at the DAC core input latch. Intuitively, increasing the delay in the reference clock output delays arrival of the data relative to the DAC clock. Conversely increasing the loop-clock delay, within the DLL/PLL’s feedback loop, has the opposite effect and advances the data timing. Only one delay should be adjusted else they will counteract each other and unnecessarily add phase noise to the clock. There is no requirement to match PCB tracking of the loop-clock signals with either the data or reference clock.

Every design implementation of FPGA, DAC and PCB needs to be evaluated to identify the correct clock delay settings. This is achieved by sending test data to the DAC and adjusting the delays, while monitoring the analog output for an uncorrupted signal. Depending on the conversion rate being used one or more valid data ‘eyes’ should be identifiable across the delay adjustment range.

During operation the loop-clock works on a similar basis to the way that the two delays counteract each other. For example, consider assembling another board using new components. Almost certainly there will be device-to-device process variations, resulting in silicon which is either faster or slower in terms of operating and propagation speeds. A consequence of this is that the propagation delays for the reference clock and data input will change but not by the same amount, thus changing the clock-to-data timing if loop-clock is not implemented. However, with loop-clock, change to the reference clock output will be largely matched to changes in the on-chip loop-clock thus automatically compensating for the device-to-device variation.

For simplicity, this explanation only considered varying one parameter but the argument can be extended to cover effects due to voltage and temperature changes in both the DAC and the key elements of the FPGA which fall within the loop-clock.

Variations in PCB propagation times, within the loop, are also compensated for but these are not expected to exhibit significant variation between boards as it will be the on-chip delays that dominate.
2 Detailed Description

This section details specific recommendations for implementing the loop-clock interface using a Xilinx Virtex-5 FPGA. The FPGA and DAC interconnections are illustrated in Figure 2.

![Figure 2. Loop-Clock implementation using a Xilinx Virtex-5 FPGA](image)

The DAC is supplied with a differential, low-jitter sinusoidal clock. Derived from this, either of the two Clock Outputs can be used to drive a Global Clock (GCLK) input on the FPGA. In this example output 1 is chosen.

Where stipulated, the use of Global Clock inputs is important. Global Clocks benefit from being routed in metal thereby avoiding significant delay variations that will otherwise occur from standard inputs routed through the FPGA fabric.

The PLL Reference Clock input must be capable of operating at the chosen DAC clock output frequency, which for 1.3GSa/s operation would normally be 650MHz. If this exceeds the PLL

Disclaimer: The contents of this document are subject to change without notice. Customers are advised to consult with FUJITSU sales representatives before ordering. The information and circuit diagrams in this document are presented "as is"; no license is granted by implication or otherwise.
specification then a divided clock output should be enabled in the DAC, for example divide by 2 to reduce the PLL Reference Clock to 325MHz.

The PLL must be capable of generating the Data Rate Clock necessary to output data at the required rate. In the above example the Data Rate Clock is 650MHz since the SERDES blocks are configured in their DDR mode.

The PLL is configured according to the Reference Clock frequency being used to generate the required Data Rate Clock. For example, either 650MHz Reference Clock x 1 = 650MHz (DDR) or 325MHz Reference Clock x 2 = 650MHz (DDR). The Data Rate Clock is used to clock the output data latches. These are implemented using fourteen 4-bit SERDES (OSERDES) blocks in parallel, driving general purpose LVDS output buffers (OBUFDS). The output data latches are 4-bit load SERDES to ensure precise data timing of the data interface to the rest of the FPGA via a 14-bit bus.

A dedicated feedback clock output from the PLL (CLKFBOUT) is used to provide the Loop-Clock. This is output via a general purpose LVDS output buffer (OBUFDS) to the Loop Clock Input on the DAC. Loop Clock Output, fed back from the DAC to the FPGA, is routed via a second Global Clock (GCLK) input to the PLL Feedback Clock input.

The Loop-Clock frequency (CLKFBIN) must be compatible with the PLL feedback input specification. In the above example 650MHz cannot be supported so the PLL Loop-Clock output is divided by 2 to operate at 325MHz. Operating the Loop-Clock at a lower frequency does not significantly detract from its principle of operation, but this should be kept as high as possible.

In the Xilinx application developed for this evaluation the data interface described above is replicated to enable data to either be routed to both ports on the DAC or to support the interleaved data mode.

The data bus between the FPGA and the DAC should be routed with matched length PCB traces to minimise skew, along with the overall trace lengths for Port A and Port B (where applicable) as the relative timing of the two interfaces is not adjustable. The clock and loop-clock traces do not need to be matched in length to the data bus as any difference is accounted for during the evaluation process.

All outputs associated with a data port should be assigned to the same I/O Bank.
3 Evaluation System

To demonstrate functionality of the loop-clock architecture an evaluation system was constructed using the Fujitsu DK86065-2 DAC demonstration kit and a Xilinx ML550 Networking Interface Platform. To simplify interconnection of the two boards, Fujitsu developed a passive interface adapter which plugs onto the data and clock interface 0.1” pitch headers of the DK86065-2. Two connectors on the adapter then connect directly to the Precision Interconnect high-speed ribbon cables provided with the ML550 kit as documented in Table 1.

<table>
<thead>
<tr>
<th>Fujitsu Adapter</th>
<th>connects to</th>
<th>Xilinx ML550</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL7</td>
<td>P74</td>
<td></td>
</tr>
<tr>
<td>SL8</td>
<td>P6</td>
<td></td>
</tr>
</tbody>
</table>

Each board is controlled via its own proprietary PC-USB interface adaptor. The complete evaluation system setup is illustrated in Figure 3.

Figure 3. Evaluation system combining Fujitsu DK86065-2 with Xilinx ML550

While Fujitsu created the correct DAC configuration settings for 1.3GSa/s operation, Xilinx developed the necessary FPGA implementation. Primarily the latter concerned the high-speed interface IP, as illustrated in Figure 2. It is this implementation that would be re-used in a customer’s application. In addition to the Interface IP, a basic data generator function was implemented in the FPGA to provide a waveform to drive the DAC. This approach allows users to evaluate the loop-clock architecture without having to either integrate the interface IP into an existing design or provide external data to the FPGA. However, both these might be considered as next-steps in any evaluation.
The FPGA data generator can accommodate waveform pattern lengths up to 65536 points, in multiples of four points. The waveform is stored in Block-RAM and is hard-coded into the FPGA design during synthesis. The Xilinx ISE design tools require the waveform to be stored in a specific format as 14 separate text [.txt] files, where each file provides a particular bit of the 14-bit pattern. A PC Windows utility was developed by Fujitsu to take a waveform pattern file, as used for the WMM by the DK86065-2 development kit software, and convert this into the required files. This utility is illustrated in Figure 4.

![Figure 4. PC utility for conversion of waveform files](image)

The .txt files generated are plain text files, formatted as four bits per line (e.g. 0101) without spaces or commas, and ordered in accordance with Table 2.

<table>
<thead>
<tr>
<th>sample 3 (bit 0)</th>
<th>sample 2 (bit 0)</th>
<th>sample 1 (bit 0)</th>
<th>sample 0 (bit 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample 7 (bit 0)</td>
<td>sample 6 (bit 0)</td>
<td>sample 5 (bit 0)</td>
<td>sample 4 (bit 0)</td>
</tr>
<tr>
<td>sample 11 (bit 0)</td>
<td>sample 10 (bit 0)</td>
<td>sample 9 (bit 0)</td>
<td>sample 8 (bit 0)</td>
</tr>
<tr>
<td>sample 15 (bit 0)</td>
<td>sample 14 (bit 0)</td>
<td>sample 13 (bit 0)</td>
<td>sample 12 (bit 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>etc.</td>
</tr>
</tbody>
</table>

The fourteen files are generated with files names ‘bit0.txt’ through to ‘bit13.txt’. Each file contains the relevant bit of every data sample, i.e. file ‘bit4.txt’ contains only bit 4 data for the entire pattern, over (number of samples/4) lines.

To define the pattern length that will be used during synthesis the parameter ‘C_MAX_NUMBER_SAMPLES’, highlighted in Figure 5, of module ‘DAC_IF_TOP behavioral .vhd’ needs to be modified. This value should be between 4 and 65536, in multiples of 4.
By default, data is routed to Port A. However, other data routing modes that can be selected are Port B only, the same data to both Port A and B, or interleaved data to Ports A and B. These alternative modes are selected during operation by using the Xilinx ChipScope Pro Analyzer tool. In the Unit 0 VIO Console window illustrated in Figure 6, for the FPGA device in the JTAG chain, signals SyncOut[0:1] determine the data routing mode in accordance with Table 3.

### Table 3: Data routing modes

<table>
<thead>
<tr>
<th>SyncOut[1]</th>
<th>SyncOut[0]</th>
<th>Data Output Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Data to Port A only (default)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Data to Port B only</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Same data to Port A and Port B</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Interleaved data through Port A and B</td>
</tr>
</tbody>
</table>

![Figure 6. Xilinx ChipScope Pro Analyzer showing FPGA VIO console window](image)
4 Results

As described in the previous sections, every new FPGA + DAC implementation needs to be evaluated to identify the appropriate delay settings for optimum Clock-to-Data timing. This applies equally to the evaluation platform described in this document as to a final application PCB layout. This process would be repeated over a number of prototype boards in order to be sure that the middle of a valid data eye has been identified. The result of this process is illustrated in Figure 7, while driving the single data port, Port A, at 1.3GSa/s.

Figure 7. Delay sweep to identify a valid data eye

In the above example, there appears to be a choice between two settings, either clkout1_clk_dly = 3 or loop_clk_dly = 7. In this instance it is recommended to select clkout1_clk_dly = 3 as the smallest delay setting, to either side of '0', is preferred.

As a comparison Figure 8, below, illustrates the extent to which clock-to-data timing changes if Loop-Clock is not enabled. This clearly shows how the valid data eye is reduced to such an extent that a single delay setting cannot be chosen.

Figure 8. Delay sweep over multiple boards without Loop-Clock enabled
5 Customer Support

General enquiries and requests for support on any aspect of Fujitsu’s high performance DACs should be e-mailed to:

msd.support@fme.fujitsu.com

For information and requests for support on any aspect of the Xilinx Virtex-5 FPGA please refer to the Xilinx website at:

http://www.xilinx.com/
Disclaimer: The contents of this document are subject to change without notice. Customers are advised to consult with FUJITSU sales representatives before ordering. The information and circuit diagrams in this document are presented "as is"; no license is granted by implication or otherwise.