High-Speed Data Interface for Stratix Devices & Fujitsu MB86064 DACs

Introduction

Implementing the digital interface to drive a high-speed digital-to-analogue converter (DAC) is challenging. The conversion rates of high-speed DACs has increased significantly in recent years, so special design techniques are required to ensure data integrity.

This application note describes a reference design that implements a high-speed data interface between a Stratix™ device and a Fujitsu MB86064 DAC. The interface comprises two 14-bit parallel buses, each running at up to 800 million samples per second (MSPS). A key feature of the reference design combines the Stratix enhanced phase-locked loop (PLL) with the MB86064 loop-clock facility to maintain optimum clock-to-data timing.

For more information on Stratix PLLs, refer to the *Stratix Device Handbook*.

For more information on the Fujitsu MB86064, refer to the Fujitsu Microelectronics Europe web site at www.fme.fujitsu.com.

Background

The Fujitsu MB86064 is a dual 14-bit 800-MSPS DAC. To achieve high data transfer rates, Stratix devices support true low-voltage differential signaling (LVDS) I/O interfaces. Each differential pair features a dedicated serializer/deserializer (SERDES) that supports transfer rates up to 840 MSPS.

Maintaining valid clock-to-data timing, particularly for a parallel bus, becomes increasingly difficult at higher clock rates. The MB86064 helps to minimize problems by implementing a double data-rate (DDR) interface at up to 400 MHz or 800 MSPS, but at this rate the clock period is only 2.5 ns with data changing every 1.25 ns. Any skew on individual outputs directly impacts the eye opening and thus increases sensitivity to clock-to-data timing.

To implement such a bus interface between two digital components, traditionally a source clock strategy is adopted. With the clock provided by the data generating device, common routing and I/O delays help maintain optimum timing to latch data at the receiver, which is not the case where data is being fed to a high performance DAC and the converter needs to be the clock master. Here the source clock is fed directly to the converter to ensure highest spectral purity, and a reference

clock output generated to synchronize the Stratix device. In its basic form this receiver generated reference clock does not benefit from the same predictability of a source generated clock and clock-to-data timing is susceptible to variations in on-chip and I/O propagations, which may result from supply variations but more typically from device-to-device variations.

A solution to reducing such an interface's susceptibility to these variations when using the MB86064 is to combine its loop-clock facility with the Stratix enhanced PLL that it uses to clock out the LVDS data. The MB86064 loop-clock is an LVDS input driving a LVDS output, through a programmable delay. By incorporating this feature, and by definition the associated tracking from the Stratix device, within the feedback loop of the PLL the resultant clock tracks variations—dynamically adjusting clock-to-data timing around a preset point. The system compensates for variations in the DAC, FPGA, or both.
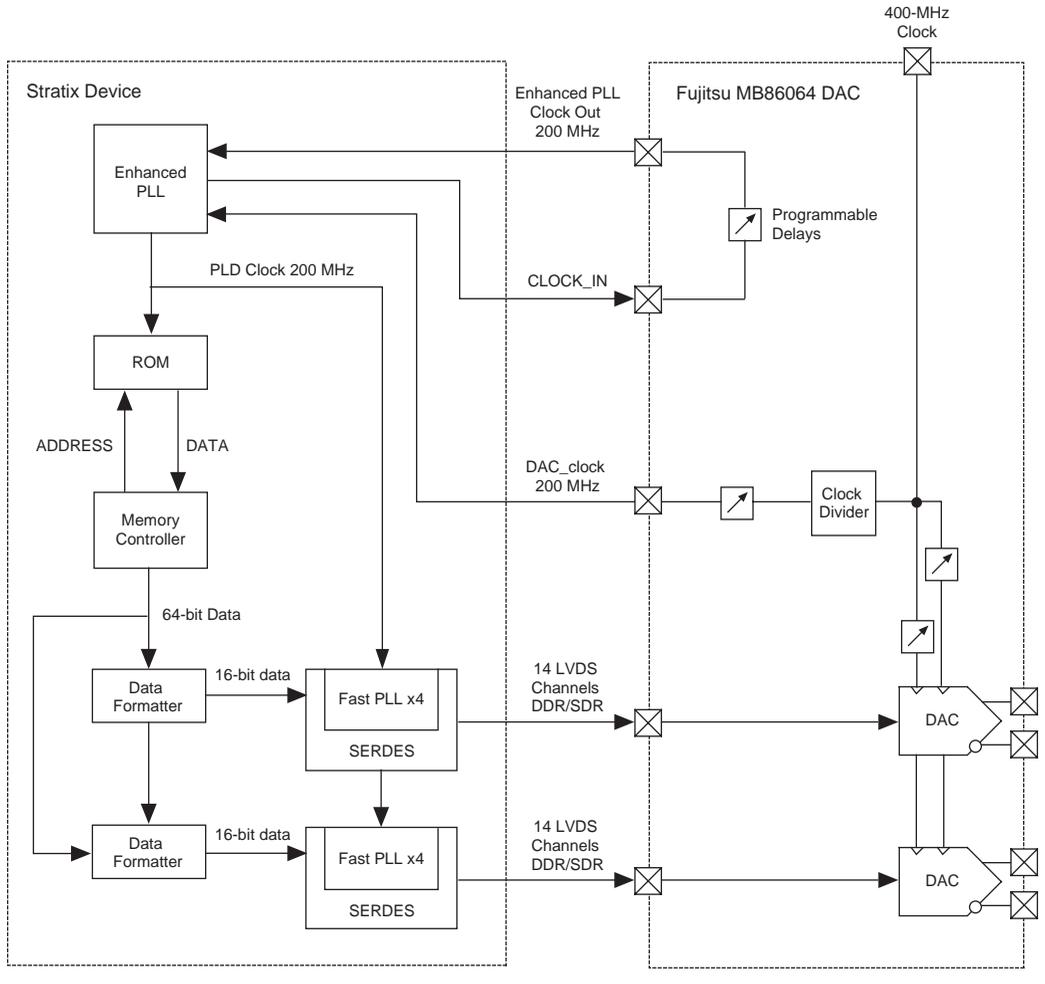
The MB86064 incorporates a number of programmable delays. Increasing the delay in the generated reference clock output retards the DAC data relative to the DAC's data latching point. Increasing the delay in the loop clock path is compensated for by the Stratix enhanced PLL so the data timing effectively advanced.

The automatic compensation of clock-to-data timing using the loop-clock is based on matching on-chip, I/O and to a lesser degree tracking delays between those in the main output clock and LVDS data paths by equivalent blocks in the PLL feedback loop. For example, if a slow DAC is substituted, resulting in a delayed output clock and thus normally retarding the clock-to-data timing, a relatively matched delay is incurred by the LVDS loop-clock output, which therefore maintains the required clock-to-data timing.

Functional Description

Figure 1 shows the reference design block diagram. The DAC derives two output clocks from the input clock and these clocks feed the enhanced PLLs on the Stratix device. Maintaining valid clock to data timing is increasingly difficult at high clock rates, particularly over tolerance. The DAC avoids potential problems through its double data rate (DDR) interface by providing a loop clock facility. The loop clock is used as a feedback clock pin on the enhanced PLL to compensate any skew or delay due to temperature or propagation through the printed circuit board (PCB).

Figure 1. Block Diagram

The system reference clock, 400 MHz, is fed directly to the DAC at half the required conversion rate, 800 MSPS, as each DAC operates double-edge clocking of the DDR input data. There are a number of programmable clock delays incorporated in the MB86064. Those relevant to the loop-clock implementation are illustrated. Each of these provides approximately 0 to 1.5 ns of delay in approximately 100 ps steps. A reference clock output is provided, which is generated through a programmable divider and its own programmable delay, and is divided by two. This clock output is used as a clock reference by the Stratix enhanced PLL. Increasing the output clock delay delays the arrival of

clock edges to the data generating device, which effectively retards the resultant clock-to-data timing at the DAC cores. This action assumes no adjustment is made to the DAC clock delay.

The clock provided by the DAC directly feeds the enhanced PLL. The output of the PLL provides the system clock that drives the entire design at 200 MHz. The design uses the internal Stratix memory to store 16 K of sine wave vectors to generate a sine wave at the output of the DAC. The memory (ROM) is read by the memory controller (up counter) and presents a 64-bit wide word to the data formatter. The data formatter re-aligns the data to the SERDES, which ensures that the 16-bit words are fed into the DAC in the right order.

The SERDES operates at four times the internal frequency, therefore one clock cycle must load 16 × 4 (64) bits. Table 5 shows the data formatter mapping. The fast PLL in the SERDES module is cascaded to the enhanced PLL providing the 800 MSPS data output from a 200 MHz system clock. Figure 2 shows the data formatter organizing and multiplexing the data.

Figure 2. Data Formatter
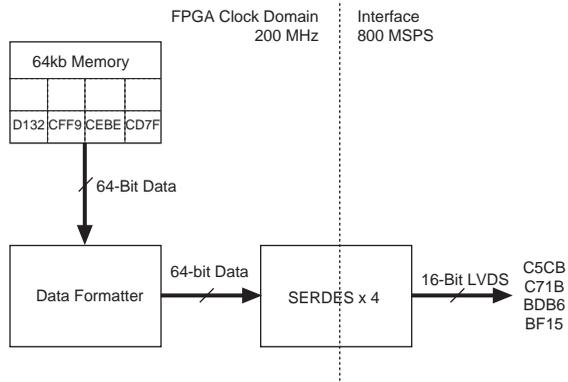
To complete the PLL implementation, the feedback clock is derived from its output clock having passed through the DAC loop-clock circuitry. At any particular setting of clock delays the system establishes the resultant clock-to-data timing. Not quite as intuitive as when increasing the output clock delay, adjusting the loop-clock delay has the opposite effect. Now the PLL compensates for the increased delay in the feedback loop and consequently advances the relative clock-to-data timing. Do not increase both the output clock delay and the loop-clock delay at the same time; they will negate each other with no resulting adjustment.

With the loop established, valid settings of output clock delay/loop-clock delay versus DAC clock delay are identified through evaluation. From these results delay settings for the design are fixed.

Figure 3 shows the Altera intellectual property (IP) used in the reference design for one DAC channel.

Figure 3. Reference Design IP for One DAC Channel
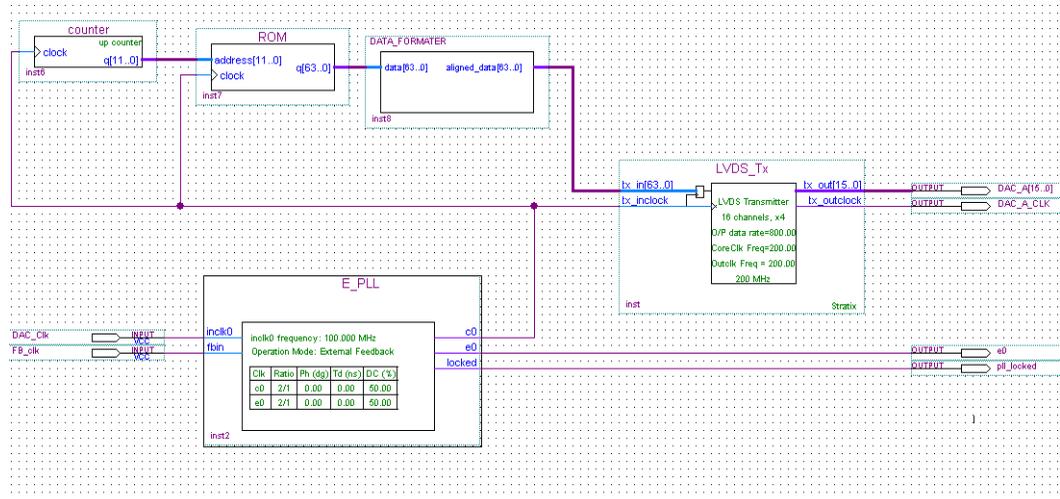
Altera's IP is used in the standard parts of the design. The dedicated RTL is provided as an example so you can be re-use it for similar applications.

The design uses the following IP:

■ Enhanced PLL
■ LVDS blocks
■ ROM
■ LPM counter
■ Custom data formatter

Table 1 shows the enhanced PLL parameters.

Table 1. Enhanced PLL Parameters	
Parameter	**Setting**
Device family	Stratix
PLL type	Enhanced
Input clock frequency	220 MHz
Operation mode	External feedback
Board level connection	e0
Create locked output	Turn on
Bandwidth setting	Preset, low
c0 core output clock	Turn on
c0 Clock multiplication factor	2
c0 Clock division factor	1
c0 Clock duty cycle	50%
e0 external output clock	Turn on
e0 Clock multiplication factor	2
e0 Clock division factor	1
e0 Clock duty cycle	50%

Table 2 shows the LVDS transmitter parameters.

Table 2. LVDS Transmitter Parameters	
Parameter	**Setting**
Device family	Stratix
Number of channels	16
Deserialization factor	4
Output divide factor	1
Output data rate	800
Alignment of data to tx_inclock	Edge aligned
Alignment of data to tx_outclock	Edge aligned
Clock frequency	200 MHz
Register input tx_inclock	Turn on
Common PLLs for receive and transmit	Turn on
tx_coreclock resource	Turn on

Table 3 shows the ROM parameters.

Table 3. ROM Parameters

Parameter	Setting
Device family	Stratix
Output bus	64 bits
Address bus	12 bits
Clock	Single
Register Q output port	Turn on

Table 4 shows the LPM counter parameters.

Table 4. LPM Counter Parameters

Parameter	Setting
Output bus width Q	2 bits.
Counter direction	Up only.
Type of counter	Plain binary.

The data formatter is a custom IP block that organizes and multiplexes the data between the 64-bit ROM interface and both 16-port SERDES channels. One clock cycle in the FPGA domain generates a 4 × 16-bit word. Table 5 shows data formatter mapping—the relationship between the ROM data and the SERDES inputs.

Table 5. Data Formatter Mapping

ROM Data Bus	SERDES Inputs Data Bus
data[0]	aligned_data[0]
data[16]	aligned_data[1]
data[32]	aligned_data[2]
data[48]	aligned_data[3]
data[1]	aligned_data[4]
data[17]	aligned_data[5]
data[33]	aligned_data[6]
data[49]	aligned_data[7]
data[2]	aligned_data[8]

Table 5. Data Formatter Mapping	
ROM Data Bus	**SERDES Inputs Data Bus**
data[18]	aligned_data[9]
data[34]	aligned_data[10]
data[50]	aligned_data[11]
data[3]	aligned_data[12]
data[19]	aligned_data[13]
data[35]	aligned_data[14]
data[51]	aligned_data[15]
data[4]	aligned_data[16]
data[20]	aligned_data[17]
data[36]	aligned_data[18]
data[52]	aligned_data[19]
data[5]	aligned_data[20]
data[21]	aligned_data[21]
data[37]	aligned_data[22]
data[53]	aligned_data[23]
data[6]	aligned_data[24]
data[22]	aligned_data[25]
data[38]	aligned_data[26]
data[54]	aligned_data[27]
data[7]	aligned_data[28]
data[23]	aligned_data[29]
data[39]	aligned_data[30]
data[55]	aligned_data[31]
data[8]	aligned_data[32]
data[24]	aligned_data[33]
data[40]	aligned_data[34]
data[56]	aligned_data[35]
data[9]	aligned_data[36]
data[25]	aligned_data[37]
data[41]	aligned_data[38]
data[57]	aligned_data[39]
data[10]	aligned_data[40]
data[26]	aligned_data[41]

Table 5. Data Formatter Mapping	
ROM Data Bus	**SERDES Inputs Data Bus**
data[42]	aligned_data[42]
data[58]	aligned_data[43]
data[11]	aligned_data[44]
data[27]	aligned_data[45]
data[43]	aligned_data[46]
data[59]	aligned_data[47]
data[12]	aligned_data[48]
data[28]	aligned_data[49]
data[44]	aligned_data[50]
data[60]	aligned_data[51]
data[13]	aligned_data[52]
data[29]	aligned_data[53]
data[45]	aligned_data[54]
data[61]	aligned_data[55]
data[14]	aligned_data[56]
data[30]	aligned_data[57]
data[46]	aligned_data[58]
data[62]	aligned_data[59]
data[15]	aligned_data[60]
data[31]	aligned_data[61]
data[47]	aligned_data[62]
data[63]	aligned_data[63]

Table 6 shows the pinout table for the Stratix device and a Fujitsu MB86064 DAC.

Table 6. Pinouts (Part 1 of 2)			
Stratix Pin	**Design Pin**	**Altera Board Pin** *(1)*	**Fujitsu Board Pin**
N24	LVDS_DATA_OUT_p0	LVDS16	NC
	LVDS_DATA_OUT_N0		
N26	LVDS_DATA_OUT_p1	LVDS15	NC
	LVDS_DATA_OUT_n1		
G23	LVDS_DATA_OUT_p2	LVDS2	D1

Table 6. Pinouts (Part 2 of 2)			
Stratix Pin	**Design Pin**	**Altera Board Pin** *(1)*	**Fujitsu Board Pin**
	LVDS_DATA_OUT_n2		
H24	LVDS_DATA_OUT_p3	LVDS3	D2
	LVDS_DATA_OUT_n3		
H22	LVDS_DATA_OUT_p4	LVDS4	D3
	LVDS_DATA_OUT_n4		
J24	LVDS_DATA_OUT_p5	LVDS5	D4
	LVDS_DATA_OUT_n5		
K23	LVDS_DATA_OUT_p6	LVDS6	D5
	LVDS_DATA_OUT_n6		
J21	LVDS_DATA_OUT_p7	LVDS7	D6
	LVDS_DATA_OUT_n7		
K21	LVDS_DATA_OUT_p8	LVDS8	D7
	LVDS_DATA_OUT_n8		
L22	LVDS_DATA_OUT_p9	LVDS9	D8
	LVDS_DATA_OUT_n9		
L23	LVDS_DATA_OUT_p10	LVDS10	D9
	LVDS_DATA_OUT_n10		
L20	LVDS_DATA_OUT_p11	LVDS11	D10
	LVDS_DATA_OUT_n11		
M22	LVDS_DATA_OUT_p12	LVDS12	D11
	LVDS_DATA_OUT_n12		
M24	LVDS_DATA_OUT_p13	LVDS13	D12
	LVDS_DATA_OUT_n13		
N20	LVDS_DATA_OUT_p14	LVDS18	D13
	LVDS_DATA_OUT_n14		
N22	LVDS_DATA_OUT_p15	LVDS17	D14
	LVDS_DATA_OUT_n15		
K17	DAC_clk	CLK14p	CLK1_OUT
H14	FB_clk	PLL5_FBp	LPCLK_OUT
L23	e0	LVDS24	LP_CLK_IN
T21	PLL_lock	LVDS10	

Notes:
(1)   Altera product engineering development board.

Getting Started

This section involves the following steps:

- "Hardware & Software Requirements"
- "Install the Design"
- "Simulate the Design"
- "Synthesize & Compile the Design"
- "Program the Stratix Device"

Hardware & Software Requirements

The reference design requires the following hardware and software:

- Custom PCB that contains a Stratix device and Fujitsu DAC
- Sine wave generator
- Quartus® II software version 3.0, or higher
- ModelSim simulator version 5.7f

☞   The design assumes that the layout of the PCB is ???????

Install the Design

To install the reference design, run **stratix_hs_dac-*<version>*.exe** and follow the installation instructions. Figure 4 shows the directory structure.

Figure 4. Directory Structure

stratix_hs_dac

  doc
  Contains the documentation.

  simulation
  Contains the testbench.

    models
    Contains models for the Altera IP.

    scripts
    Contains the simulation scripts.

  source
  Contains the source files.

    ip
    Contains the Verilog HDL source files.

    quartus
    Contains the files for the Quartus II software and SOPC Builder.

    software
    Contains the software source files.
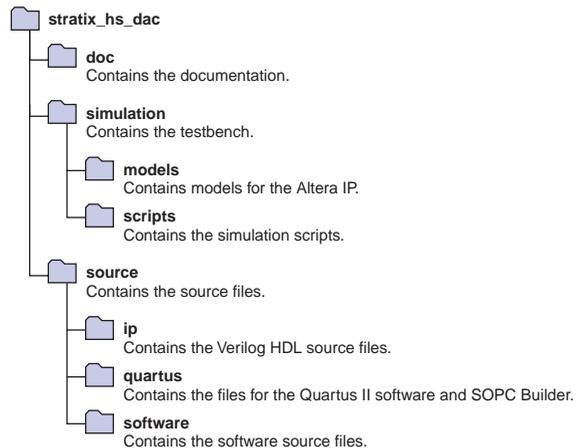
Table 7 shows the design files.

Table 7. Design Files

Filename	Description
mb86064.quartus	Quartus II project.
pll.v	Verilog HDL and enhanced PLL configuration.
mb86064.v	Top-level design file.
LVDS.v	LVDS IP block.
ROM.v	ROM IP block.
counter.v	Counter IP block.
mb86064.csf	Pinout file.

Run the Software

An executable **mif_generator.exe** in the **\source\software\mif_generator\debug** directory generates a **.mif** file that is recognised by the Quartus II software. When you open the **.mif** file in the Quartus II software, you can save it as an **.hex** file. You must convert the file to the **.hex** format, because ModelSim uses the **converthex2ver.dll** that takes a hexadecimal input format and converts it to its proprietary format to initialize the memory models.

The source code for the **mif_generator.exe** is provided so you can modify it if you wish. Different waveforms of different sizes can be generated from changing the **sinus.cpp** file and re compiling it using for example Microsoft Visual C++.

In the **\source\software\mif_generator\debug** directory, type the following command line, to run the executable.

```
mif_generator <file_name>.mif
```

Simulate the Design

The design includes a testbench to simulate the design using the ModelSim simulator. To use simulate the design, perform the following steps:

1. Copy the **convert_hex2ver.dll** file from the **dac\quartus** directory to the *<modeltech installation>***win32** directory.

2. Edit your **modelsim.ini** file:

a.   Go to the following section:

```
; List of dynamically loaded objects for Verilog PLI
applications
```

b.   Add the following line:

```
Veriuser = convert_hex2ver.dll
```

3.   Start the ModelSim simulator and change to the **simulation/scripts** directory.

4.   Run the **macro.do** file to compile and display all the relevant waveforms (see Figure 5).

Figure 5. Example ModelSim Waveform
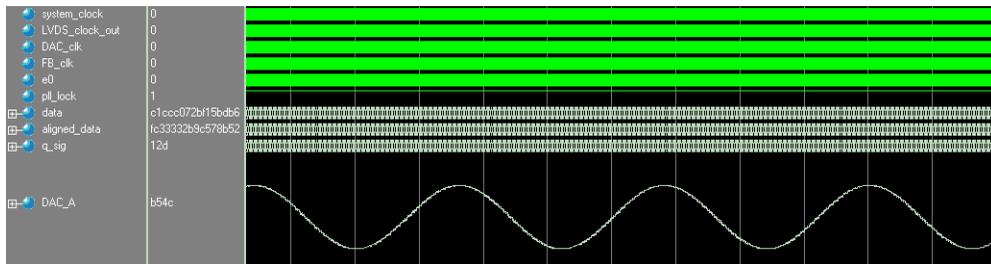
The simulator converts the vector file for the memory (ROM) using the **convert_hex2ver dll**, in this design **sine_wave.hex** to **sine_wave.ver**.

If you wish to change that vector file, the design includes some C code to generate new waveforms with as many vectors as the memory can hold (see "Run the Software" on page 12).

Synthesize & Compile the Design

To synthesise and compile the design, perform the following steps:

1.   Start the Quartus III software.

2.   Open the Quartus II project **source\quartus\mb86064.quartus**

3.   Choose **Start Compilation** (Processing menu).

All the design's IP are set up using the MegaWizard Plug-In, if any you want to change any parameters perform the following steps:

1. Open the appropriate MegaWizard Plug-In, by choosing **MegaWizard Plug-In Manager** (Tools menu).

2. Change the setttings.

3. Re-compile the design.

Program the Stratix Device

To program the Stratix device, perform the following steps:

1. Choose **Programmer** (Tools menu).

2. Click the Auto Detect icon, to scan the JTAG chain.

3. Click the Add File icon.

4. Choose **mb86064.sof**, and click **Open**.

5. Click the Start Programming icon.

Program the DAC

Adjustments to the MB86064 programmable delays are made through a 4-wire serial control interface. For evaluation purposes these adjustments are best performed using the Fujitsu PC serial interface lead and software utility, but ultimately control and configuration of the DAC is implemented within the Stratix device or an embedded system controller as appropriate.

Performance

Figure 6 shows the test setup.

Figure 6. Test Setup
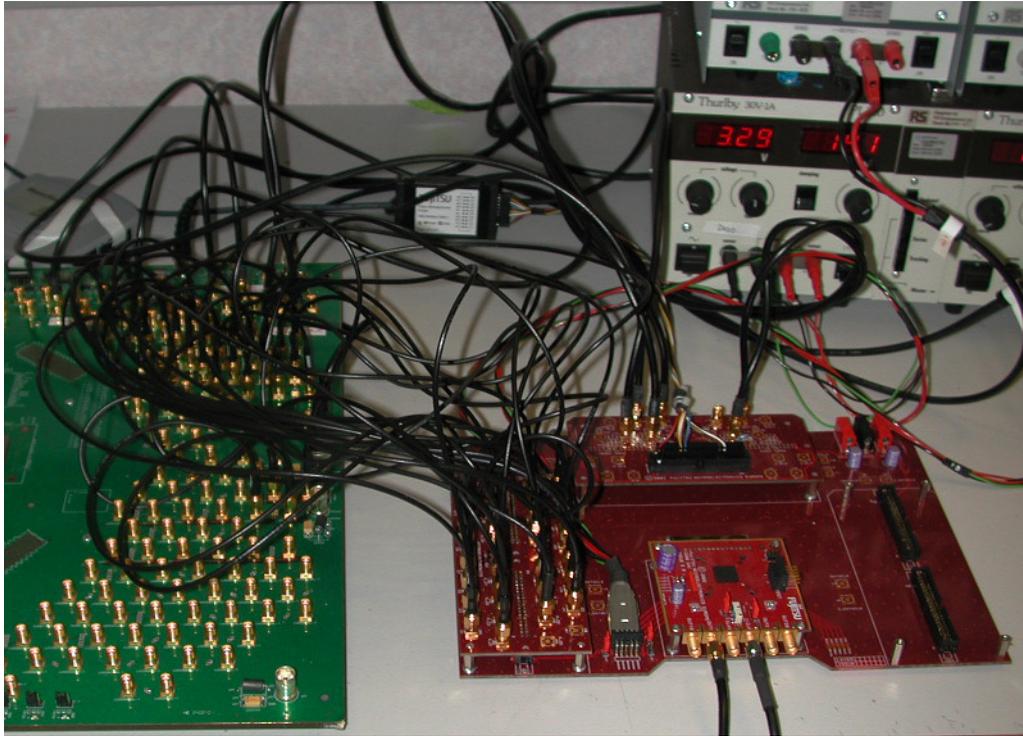
After power-up the DAC needs programming to enable the specific functions and configuration required. In this application example only one DAC channel, DAC A, is driven. The following registers need to be programmed:

- `WMM Config [0x00]` with 0x0000001, which sets clock output 1 to clock/2
- `DAC Config [0x1C0]` with 0x0000002, which sets references for single DAC A mode
- `Power Down [0x1C3]` with 0x0000B80, which powers up DAC A

The on-chip programmable delays also need to be adjusted to ensure input data is latched at the centre of the data eye. This adjustment is achieved by setting the DAC core clock delays in combination with the system clock delays to advance or retard the input data:

- DAC core clock delays [0x1B2] with 0x0000000. 0x0000011 to 0x00000FF adjusts the DAC core timing

■ System clock delays [0x1C1] with 0x0000000. 0x00000F0 to 0x0000000 retards the clock output 1; 0x0000000 to 0x0000F00 advances the loop clock delay

Combining these delays forms a 2-dimensional array of adjustments (see Figure 7). In Figure 7 the shaded area corresponds to valid data timing. The position of the permissible data eyes varies from design to design but should follow the same basic relationship.

Figure 7. Two-Dimensional Array
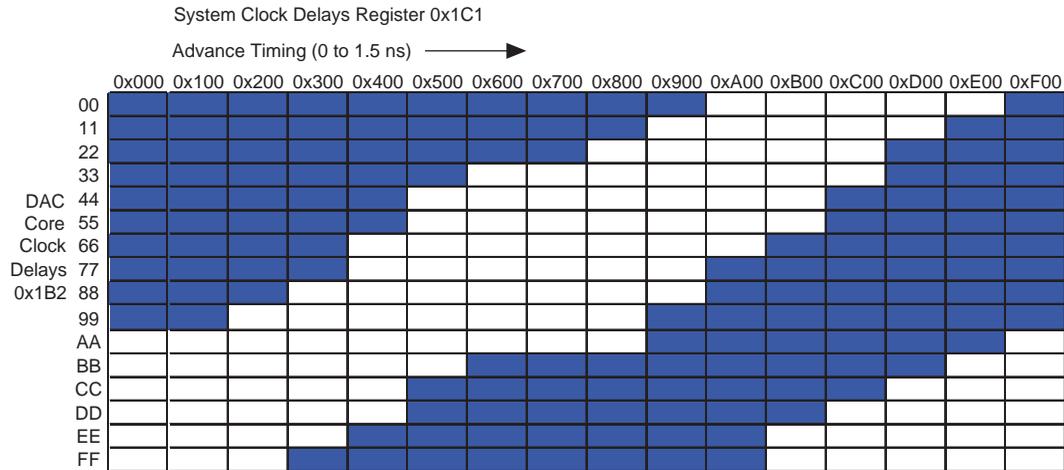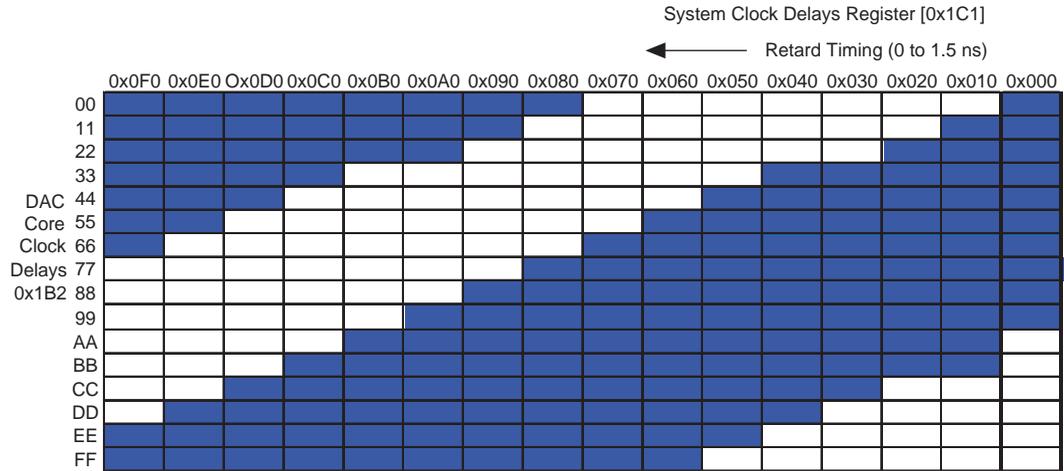
Fujitsu recommends selecting delay settings as near to the centre top where the DAC core clock delays are minimized and advance or retard.

☞   You can set the clock core delay first and adjust the advance or retard accordingly or vice versa.

The implementation of a loop-clock facility is designed to help maintain clock-to-data timing in particular from one board to the next due to device-to-device variations. To simulate the design using the evaluation platform, the power supplies to the FPGA and DAC were varied together to simulate fast and slow process variations. The worst case scenario undoubtedly being chancing upon either a both fast or both slow FPGA & DAC combination.

Figure 8 shows the results that compare using a loop-clock facility to without using a loop-clock facility. Both results are taken at a fixed advance and retard setting while the DAC core delays are adjusted to find the resultant data eye. Without a loop-clock facility, the required setting is seen to vary considerably from one configuration to the next making it impossible for a single delay setting to be reliably selected during the design evaluation stage. By contrast with the loop-clock facility implementation a valid data eye is maintained at more than one set of register settings across different devices [supply variations].

Figure 8. Results

	Low	Typ.	High
FPGA Core	1.4V	1.5V	1.6V
IOs	3.1V	3.3V	3.5V
DAC Core	1.6V	1.8V	2.0V
IOs	2.9V	3.3V	3.6V
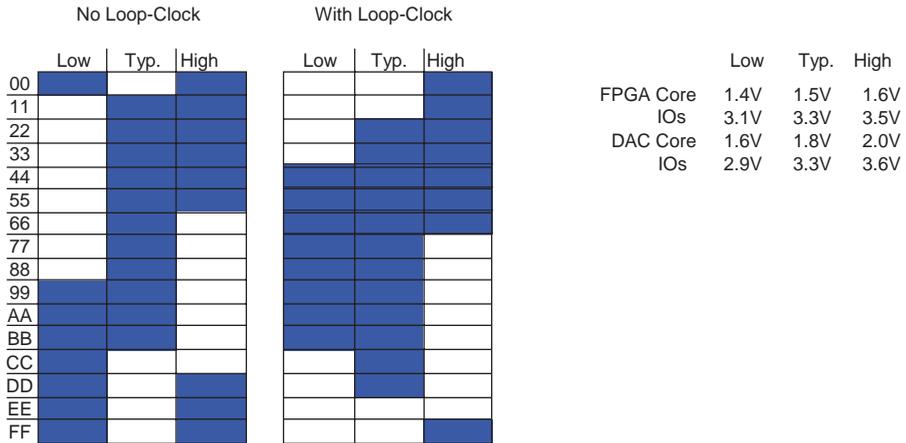
Conclusion

Congratulations! You have successfully implemented a high-speed data interface between a Stratix device and a Fujitsu MB86064 DAC, which shows that the loop-clock facility maintains clock-to-data timing in particular from one board to the next due to device-to-device variations. The design is implemented using simple wizards with a minimal amount of logic elements because of the hard-coded SERDES. The design was implemented and qualified successfully in conjuction with Fujitsu.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
lit_req@altera.com

I.S. EN ISO 9001