Using DDR for Fusion Devices

Table of Contents

Introduction

The I/Os on the Fusion device families support Double Data Rate (DDR) mode. In this mode, new data is present on every transition (or clock edge) of the clock signal. This mode doubles the data transfer rate when compared with Single Data Rate (SDR) mode where new data is present on one transition (or clock edge) of the clock signal. The Fusion families have DDR circuitry built into the I/O tiles. I/Os are configured to be DDR receivers or transmitters by instantiating the appropriate special macros and buffers (DDR_OUT or DDR_REG) in the RTL design. This application note discusses the options you can choose to configure the I/Os in this mode and how to instantiate them in the design.

I/O Cell Architecture

The Fusion families support DDR in the I/O cells in four different modes: Inputs, Outputs, Tristate, and Bidirectional pins. For each mode, different I/O standards are supported, with most I/O standards having special sub-options. Refer to Table 1 on page 2 for a sample of the available I/O options. Additional I/O options can be found in the *Fusion Family of Mixed Signal FPGAs* datasheet.

Table 1 • **DDR I/O Options**

DDR Register Type	I/O Type	I/O Standard	Sub-options	Comments
Receive Register	Input	Normal	None	3.3 V TTL (default)
		LVCMOS	Voltage	1.5 V, 1.8 V, 2.5 V, 5 V (1.5 V default)
			Pull-up	None (default)
		PCI/PCIX	None	
		GTL/GTLP	Voltage	2.5 V, 3.3 V (3.3 V default)
		HSTL	Class	I / II (I default)
		SSTL2/SSTL3	Class	I / II (I default)
		LVPECL	None	
		LVDS	None	
Transmit Register	Output	Normal	None	3.3 V TTL (default)
		LVTTL	Output drive	2, 4, 6, 8,12,16, 24, 36 (8 mA default)
			Slew rate	Low/High (High default)
		LVCMOS	Voltage	1.5 V, 1.8 V, 2.5 V, 5 V (1.5 V default)
		PCI/PCIX	None	
		GTL/GTLP	Voltage	1.8 V, 2.5 V, 3.3 V (3.3 V default)
		HSTL	Class	I / II (I default)
		SSTL2/SSTL3	Class	I / II (I default)
		LVPECL	None	
		LVDS	None	
	Tristate Buffer	Normal	Enable polarity	Low/high (low default)
		LVTTL	Output Drive	2, 4, 6, 8, 12,16, 24, 36 (8 mA default)
			Slew rate	Low/high (high default)
			Enable polarity	Low/high (low default)
			Pull-up/down	None (default)
		LVCMOS	Voltage	1.5 V, 1.8 V, 2.5 V, 5 V (1.5 V default)
			Output drive	2, 4, 6, 8, 12, 16, 24, 36. (8 mA default)
			Slew rate	Low/high (high default)
			Enable polarity	Low/high (low default)
			Pull-up/down	None (default)
		PCI/PCI-X	Enable polarity	Low/high (low default)
		GTL/GTLP	Voltage	1.8 V, 2.5 V, 3.3 V (3.3 V default)
			Enable polarity	Low/high (low default)
		HSTL	Class	I / II (I default)
			Enable polarity	Low/high (low default)
		SSTL2/SSTL3	Class	I / II (I default)
			Enable polarity	Low/high (low default)

Table 1 • **DDR I/O Options (continued)**

DDR Register Type	I/O Type	I/O Standard	Sub-options	Comments
Transmit Register (continued)	Bidirectional Buffer	Normal	Enable polarity	Low/high (low default)
		LVTTL	Output drive	2, 4, 6, 8, 12, 16, 24, 36 (8 mA default)
			Slew rate	Low/high (high default)
			Enable polarity	Low/high (low default)
			Pull-up/down	None (default)
		LVCMOS	Voltage	1.5 V, 1.8 V, 2.5 V, 5 V (1.5 V default)
			Enable polarity	Low/high (low default)
			Pull-up	None (default)
		PCI/PCIX	None	
			Enable polarity	Low/high (low default)
		GTL/GTLP	Voltage	1.8 V, 2.5 V, 3.3 V (3.3 V default)
			Enable polarity	Low/high (low default)
		HSTL	Class	I / II (I default)
			Enable polarity	Low/high (low default)
		SSTL2/SSTL3	Class	I / II (I default)
			Enable polarity	Low/high (low default)

Instantiating DDR Registers

Instantiations

Using SmartGen is the simplest way to generate the appropriate RTL files for use in the design. SmartGen provides the capability to generate all of the DDR I/O cells as described. Through the graphical user interface (GUI), you can select from among the many supported I/O standards. The output formats supported are Verilog, VHDL, or EDIF files.

Figure 2 on page 4, Figure 3 on page 5, Figure 4 on page 7, and Figure 5 on page 8 show the I/O cell configured for DDR using SSTL2 Class I technology. For each I/O standard, the I/O pad is buffered by a special primitive that indicates the I/O standard type.

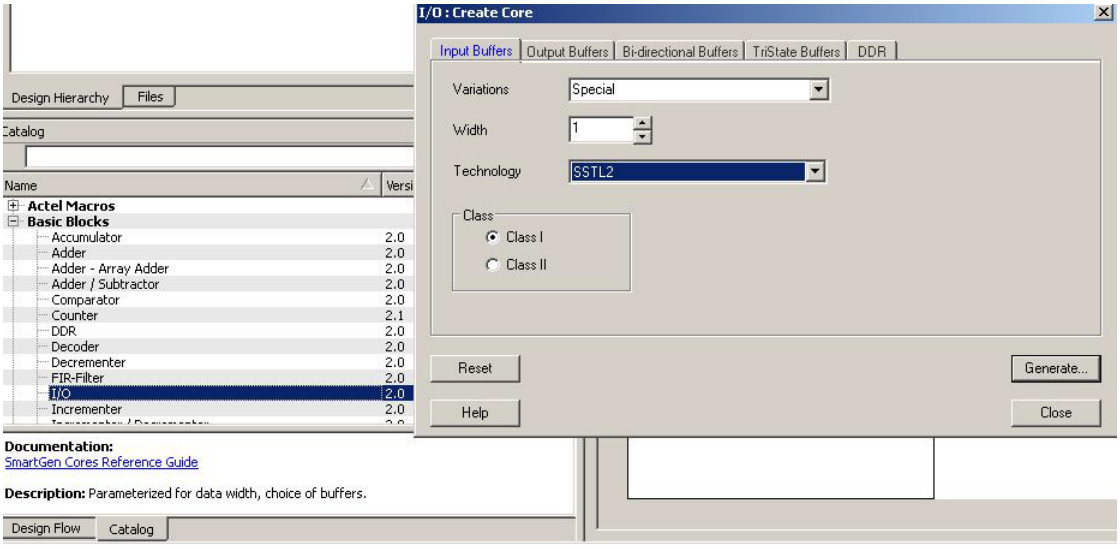

***Figure 1* • Example of Using SmartGen to Generate a DDR SSTL2 Class I Input Register**

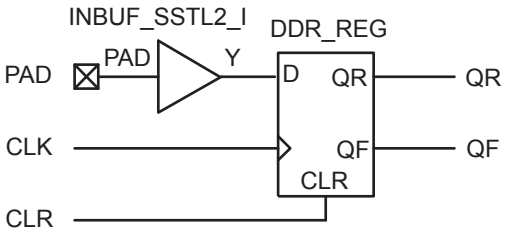

***Figure 2* • DDR Input Register (SSTL2 Class I)**

The corresponding structural representations as generated by SmartGen are shown below:

Verilog

```
module DDR_InBuf_SSTL2_I(PAD,CLR,CLK,QR,QF);

input   PAD, CLR, CLK;
output  QR, QF;

wire Y;

    INBUF_SSTL2_I INBUF_SSTL2_I_0_inst(.PAD(PAD),.Y(Y));
    DDR_REG DDR_REG_0_inst(.D(Y),.CLK(CLK),.CLR(CLR),.QR(QR),.QF(QF));

endmodule
```

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
library fusion;

entity DDR_InBuf_SSTL2_I is
    port(PAD, CLR, CLK : in std_logic;  QR, QF : out std_logic) ;
end DDR_InBuf_SSTL2_I;


architecture DEF_ARCH of  DDR_InBuf_SSTL2_I is

    component INBUF_SSTL2_I
        port(PAD : in std_logic := 'U'; Y : out std_logic) ;
    end component;

    component DDR_REG
        port(D, CLK, CLR : in std_logic := 'U'; QR, QF : out std_logic) ;
    end component;

signal Y : std_logic ;

begin

    INBUF_SSTL2_I_0_inst : INBUF_SSTL2_I
    port map(PAD => PAD, Y => Y);
    DDR_REG_0_inst : DDR_REG
    port map(D => Y, CLK => CLK, CLR => CLR, QR => QR, QF => QF);

end DEF_ARCH;
```

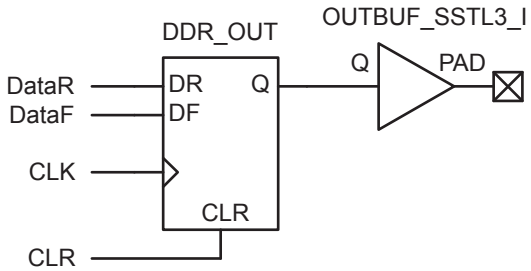

Figure 3 • **DDR Output Register (SSTL3 Class I)**

Verilog

```
module DDR_OutBuf_SSTL3_I(DataR,DataF,CLR,CLK,PAD);

input   DataR, DataF, CLR, CLK;
output  PAD;

wire Q, VCC;

    VCC VCC_1_net(.Y(VCC));
    DDR_OUT DDR_OUT_0_inst(.DR(DataR),.DF(DataF),.CLK(CLK),.CLR(CLR),.Q(Q));
    OUTBUF_SSTL3_I OUTBUF_SSTL3_I_0_inst(.D(Q),.PAD(PAD));

endmodule
```

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
library fusion;

entity DDR_OutBuf_SSTL3_I is
    port(DataR, DataF, CLR, CLK : in std_logic;  PAD : out std_logic) ;
end DDR_OutBuf_SSTL3_I;

architecture DEF_ARCH of  DDR_OutBuf_SSTL3_I is

    component DDR_OUT
        port(DR, DF, CLK, CLR : in std_logic := 'U'; Q : out std_logic) ;
    end component;

    component OUTBUF_SSTL3_I
        port(D : in std_logic := 'U'; PAD : out std_logic) ;
    end component;

    component VCC
        port( Y : out std_logic);
    end component;

signal Q, VCC_1_net : std_logic ;

begin

    VCC_2_net : VCC port map(Y => VCC_1_net);
    DDR_OUT_0_inst : DDR_OUT
    port map(DR => DataR, DF => DataF, CLK => CLK, CLR => CLR, Q => Q);
    OUTBUF_SSTL3_I_0_inst : OUTBUF_SSTL3_I
    port map(D => Q, PAD => PAD);
end DEF_ARCH;
```

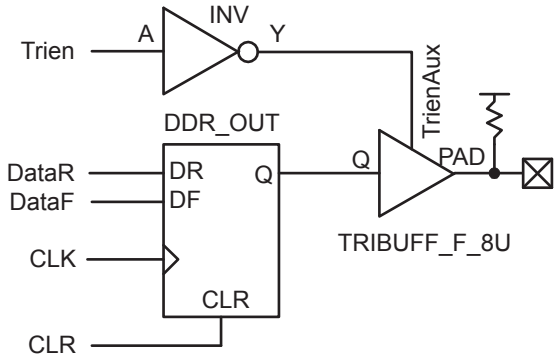

Figure 4 • **DDR Tristate Output Register, Low Enable, 8 mA, Pull-Up (LVTTL)**

Verilog

```
module DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp(DataR,
        DataF,CLR,CLK,Trien,PAD);
input   DataR, DataF, CLR, CLK, Trien;
output  PAD;

wire TrienAux, Q;

    INV Inv_Tri(.A(Trien),.Y(TrienAux));
    DDR_OUT DDR_OUT_0_inst(.DR(DataR),.DF(DataF),.CLK(CLK),.CLR(CLR),.Q(Q));
    TRIBUFF_F_8U TRIBUFF_F_8U_0_inst(.D(Q),.E(TrienAux),.PAD(PAD));

endmodule
```

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
library fusion;

entity DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp is
    port(DataR, DataF, CLR, CLK, Trien : in std_logic;  PAD : out std_logic) ;
end DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp;

architecture DEF_ARCH of
        DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp is

    component INV
        port(A : in std_logic := 'U'; Y : out std_logic) ;
    end component;

    component DDR_OUT
        port(DR, DF, CLK, CLR : in std_logic := 'U'; Q : out std_logic) ;
    end component;

    component TRIBUFF_F_8U
        port(D, E : in std_logic := 'U'; PAD : out std_logic) ;
    end component;

signal TrienAux, Q : std_logic ;

begin

    Inv_Tri : INV
    port map(A => Trien, Y => TrienAux);
```

```
    DDR_OUT_0_inst : DDR_OUT
    port map(DR => DataR, DF => DataF, CLK => CLK, CLR => CLR, Q => Q);
    TRIBUFF_F_8U_0_inst : TRIBUFF_F_8U
    port map(D => Q, E => TrienAux, PAD => PAD);

end DEF_ARCH;
```

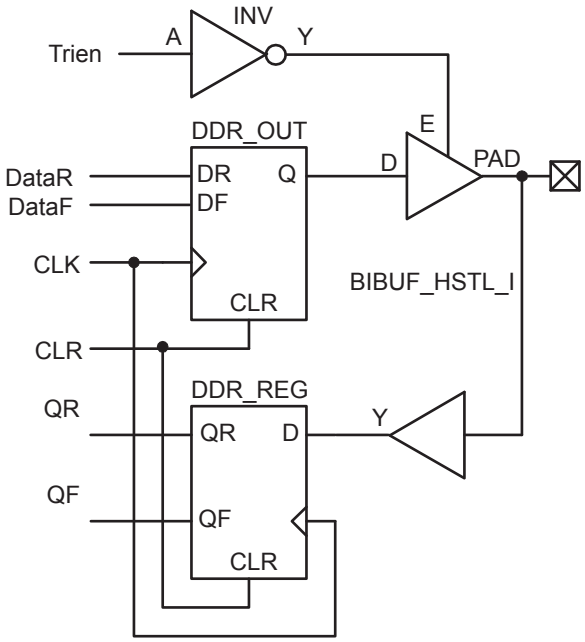

Figure 5 • **DDR Bidirectional Buffer, Low Output Enable (HSTL Class 2)**

Verilog

```
module DDR_BiDir_HSTL_I_LowEnb(DataR,DataF,CLR,CLK,Trien,QR,QF,PAD);

input   DataR, DataF, CLR, CLK, Trien;
output  QR, QF;
inout   PAD;

    wire TrienAux, D, Q;

    INV Inv_Tri(.A(Trien), .Y(TrienAux));
    DDR_OUT DDR_OUT_0_inst(.DR(DataR),.DF(DataF),.CLK(CLK),.CLR(CLR),.Q(Q));
    DDR_REG DDR_REG_0_inst(.D(D),.CLK(CLK),.CLR(CLR),.QR(QR),.QF(QF));
    BIBUF_HSTL_I BIBUF_HSTL_I_0_inst(.PAD(PAD),.D(Q),.E(TrienAux),.Y(D));

endmodule
```

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
library fusion;

entity DDR_BiDir_HSTL_I_LowEnb is
    port(DataR, DataF, CLR, CLK, Trien : in std_logic;
         QR, QF : out std_logic;  PAD : inout std_logic) ;
end DDR_BiDir_HSTL_I_LowEnb;

architecture DEF_ARCH of  DDR_BiDir_HSTL_I_LowEnb is

    component INV
        port(A : in std_logic := 'U'; Y : out std_logic) ;
    end component;

    component DDR_OUT
        port(DR, DF, CLK, CLR : in std_logic := 'U'; Q : out std_logic) ;
    end component;

    component DDR_REG
        port(D, CLK, CLR : in std_logic := 'U'; QR, QF : out std_logic) ;
    end component;

    component BIBUF_HSTL_I
        port(PAD : inout std_logic := 'U'; D, E : in std_logic := 'U';
               Y : out std_logic) ;
    end component;

signal TrienAux, D, Q : std_logic ;

begin

    Inv_Tri : INV
    port map(A => Trien, Y => TrienAux);
    DDR_OUT_0_inst : DDR_OUT
    port map(DR => DataR, DF => DataF, CLK => CLK, CLR => CLR, Q => Q);
    DDR_REG_0_inst : DDR_REG
    port map(D => D, CLK => CLK, CLR => CLR, QR => QR, QF => QF);
    BIBUF_HSTL_I_0_inst : BIBUF_HSTL_I
    port map(PAD => PAD, D => Q, E => TrienAux, Y => D);

end DEF_ARCH;
```

Design Example

Figure 6 shows a simple example of a design using both DDR input and DDR output registers. You can copy the HDL code in Libero SoC software and go through the design flow. Figure 7 and Figure 8 on page 11 show the netlist and ChipPlanner view of the ddr_test design.

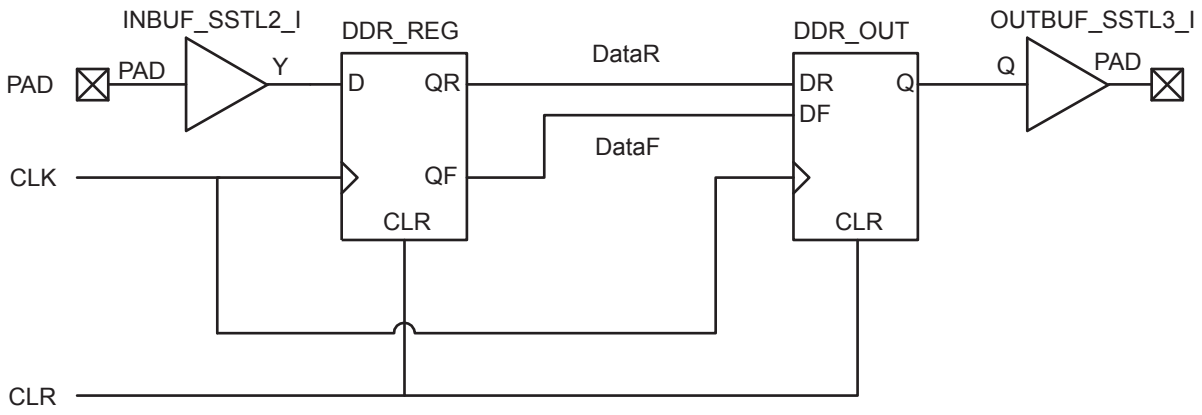

***Figure 6* • Design Example**

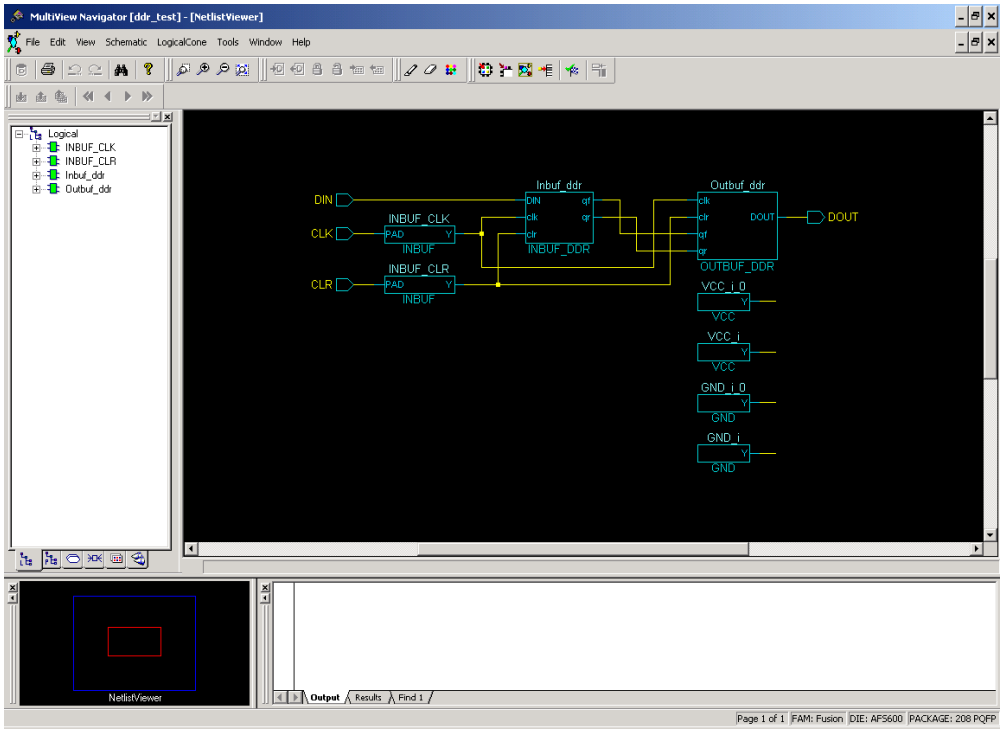

***Figure 7* • DDR Test Design as Seen by NetlistViewer**

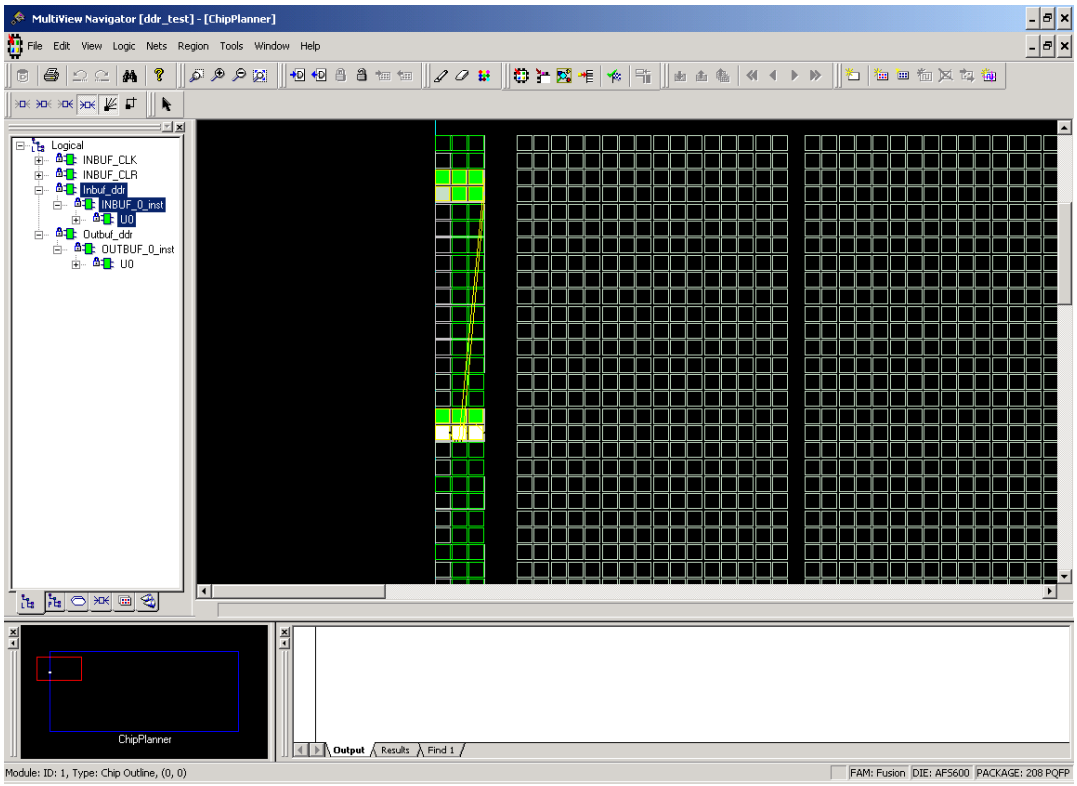

Figure 8 • **DDR Input/Output Cells as Seen by ChipPlanner**

Verilog

```
module Inbuf_ddr(PAD,CLR,CLK,QR,QF);
input PAD, CLR, CLK;
output  QR, QF;
    wire Y;
    DDR_REG DDR_REG_0_inst(.D(Y), .CLK(CLK), .CLR(CLR), .QR(QR), .QF(QF));
    INBUF INBUF_0_inst(.PAD(PAD), .Y(Y));
endmodule

module Outbuf_ddr(DataR,DataF,CLR,CLK,PAD);
input DataR, DataF, CLR, CLK;
output  PAD;
    wire Q, VCC;
    VCC VCC_1_net(.Y(VCC));
    DDR_OUT DDR_OUT_0_inst(.DR(DataR), .DF(DataF), .CLK(CLK), .CLR(CLR), .Q(Q));
    OUTBUF OUTBUF_0_inst(.D(Q), .PAD(PAD));
endmodule

module ddr_test(DIN, CLK, CLR, DOUT);

input  DIN, CLK, CLR;
output DOUT;

Inbuf_ddr    Inbuf_ddr    (.PAD(DIN), .CLR(clr), .CLK(clk), .QR(qr), .QF(qf));
Outbuf_ddr   Outbuf_ddr   (.DataR(qr),.DataF(qf), .CLR(clr), .CLK(clk),.PAD(DOUT));

INBUF        INBUF_CLR    (.PAD(CLR), .Y(clr));
INBUF        INBUF_CLK    (.PAD(CLK), .Y(clk));

endmodule
```

Simulation Consideration

Microsemi DDR simulation models use inertial delay modeling by default (versus transport delay modeling). As such, pulses that are shorter than the actual gate delays should be avoided as they are not seen by the simulator and may be an issue in post-routed simulations. You should be aware of the default delay modeling and must set the correct delay model in the simulator as needed.

Conclusion

The Fusion devices support a wide range of DDR applications with the different I/O standards and include built-in DDR macros. The powerful capabilities provided by SmartGen macro builder simplify the process of including DDR macros in the designs and minimize the design errors. The designer should take the additional considerations into account in design floorplaning and placement of I/O flip-flops to minimize the datapath skew and to help improve the system timing margins. Other system-related issues to consider include PLL and clock partitioning.

Related Documents

Datasheets

Fusion Family of Mixed Signal FPGAs

http://www.microsemi.com/soc/documents/Fusion_DS.pdf

List of Changes

The following table lists critical changes that were made in each revision of the document.

Revision*	Changes	Page
Revision 1 (February 2012)	The "Design Example" section was revised. (SAR - 32657)	10
	The "Related Documents" section was revised. (SAR - 32657)	12
	The Figure 1 was updated.	4

*Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.*

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

51900117-1/02.12