XAPP1208 (v1.0) May 16, 2014

Bitslip in Logic

Author: Marc Defossez

Summary

I/O logic in UltraScale™ devices refers to the dedicated I/O handling components located between the I/O buffers and the general interconnect. This I/O logic is different in UltraScale devices compared to that of previous families such as the 7 series and Virtex®-6 FPGAs. The I/O logic setup in UltraScale devices provides faster I/O handling, better jitter specifications, and more functionality. However, it omits some functionality available in the I/O logic of previous device families.

Bitslip is a function that is not natively available in UltraScale device I/O logic. This application note describes a Bitslip solution implemented in general interconnect that can be used in UltraScale device components as well as in previous device architectures. The reference design implements the Bitslip function and extends the basic functionality with several extra options.

Using the basic UltraScale device BITSLICE I/O primitives is referred to as “native mode” while the I/O logic functions of previous device families are mimicked using UltraScale device I/O with “component mode” primitives.

Introduction

The Bitslip function natively available in each ISERDES of previous device families acted on the serial input stream. The ISERDES equivalent (component mode) or native RX_BITSLICE function in UltraScale devices has no Bitslip functionality implemented.

This application note describes the Bitslip functionality supported natively in previous device families and how an equivalent Bitslip can be implemented for UltraScale devices. The reference design provides the described solution as a ready-to-use design that can be modified if necessary.

The provided reference design can be used with component mode (ISERDES) or native mode (RX_BITSLICE) with 4-bit and 8-bit outputs and can be customized using “attributes/generics” in the VHDL code.

The reference design can also be used in 7 series and Virtex-6 FPGA designs, which can make retargeting of designs easier. Modify the original design to use the general interconnect implemented Bitslip instead of the native Bitslip. After the design is tested in the original device technology, port it into the UltraScale device.

The component block of the Bitslip functionality as implemented in logic is shown in Figure 1. This block and its pins and attributes are explained in detail later in this document.

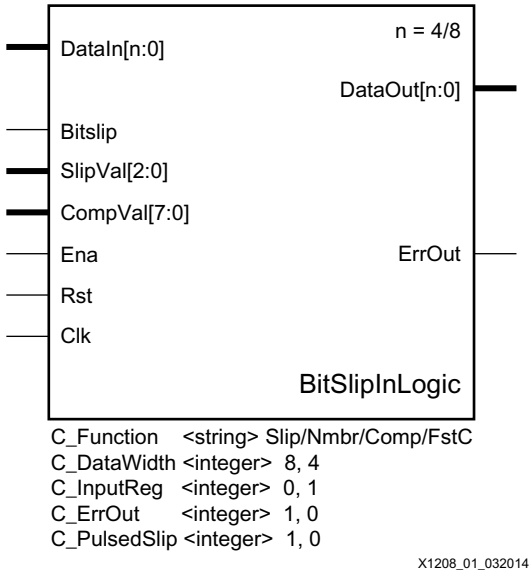


Figure 1: **Bitslip Function Block**

Native Bitslip Function in Previous Architectures

The number of bits captured is set by the DATA_WIDTH attribute when instantiating an ISERDES component in the design source code. Bits are captured in the ISERDES on the CLK clock, and the ISERDES parallel output is available in the general interconnect using the CLKDIV clock.

CLKDIV is thus a divided version of CLK. When DATA_WIDTH is set to eight and the ISERDES is used in Single Data Rate (SDR) mode, CLKDIV is CLK divided by eight. When the ISERDES is used in Dual Data Rate (DDR) mode, CLKDIV is CLK divided by four. Assuming DATA_WIDTH is set to eight:

- Bits are captured at CLK rate into an input serial-to-parallel register of the ISERDES.
- A state machine in the ISERDES running at the same CLK generates a clock pulse to capture the bits from the serial-to-parallel register in an internal register each time it reaches the value set by DATA_WIDTH. In this case, the state machine generates a clock pulse for the internal register when eight bits are captured in the serial-to-parallel register.
- The internal state machine generated clock is the equivalent of the external supplied CLKDIV clock. This internal generated clock is related to the CLK clock and is not phase aligned with the external applied CLKDIV.
- The data is transferred from the internal register to the parallel output register Q by the external applied CLKDIV clock.

A functional example is shown in Figure 2 (SDR operation). Data capture starts with bit value 7. Bits are shifted into the serial-to-parallel register at the CLK clock rate. The vertical stacked blocks represent that register. These blocks show that the bit of value 7 is shifted in first and then ends at the bottom. The last bit shifted in is the bit with value D. The internal register captures the contents from the serial-to-parallel register when it contains eight captured bits. At the next CLKDIV rising edge, the content moves to the parallel output register. That register then contains the value `DCBA0987`.

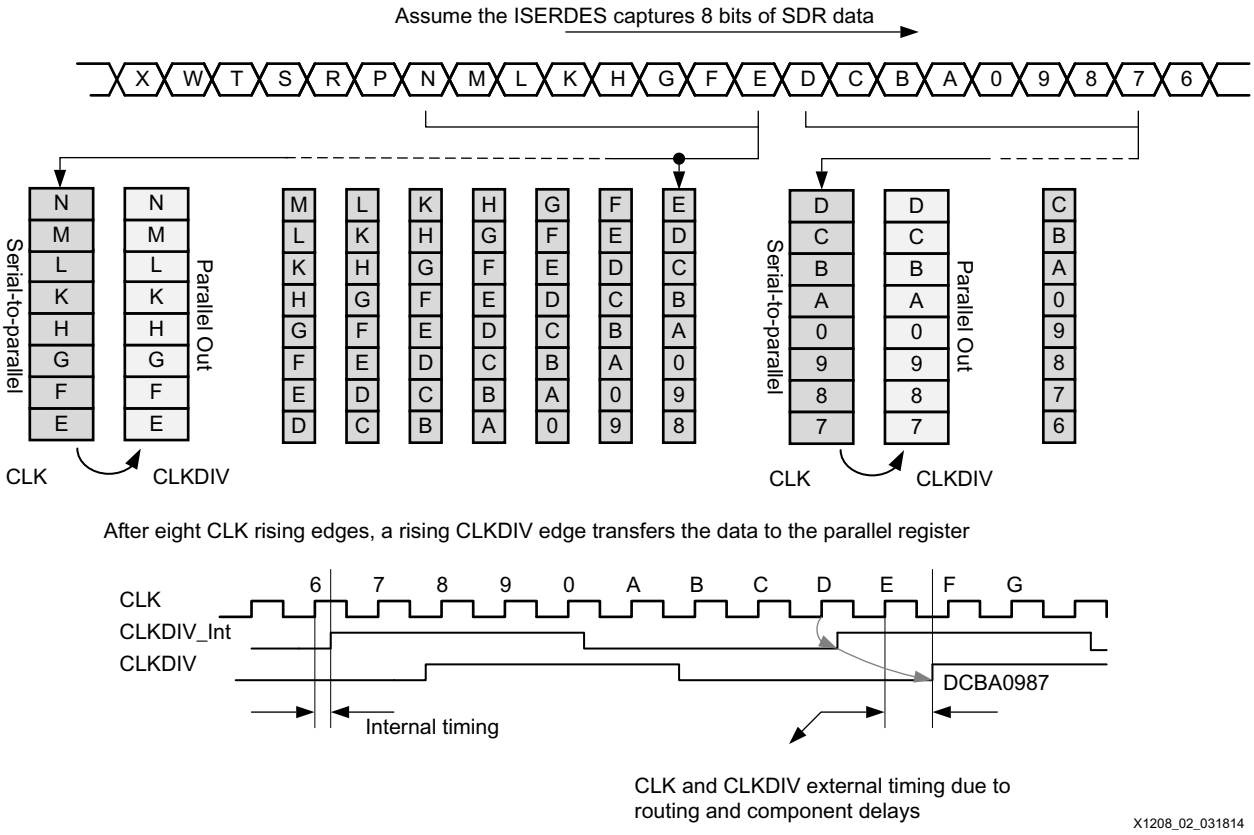


Figure 2: **Capturing Bits Without Bitslip**

As soon as data is transferred from the serial-to-parallel register into the internal storage register, consecutive new data is shifted in the serial-to-parallel register.

When the Bitslip functionality is used, the capture, transfer to internal register, and transfer to output register operates in the same manner. When the ISERDES Bitslip input signal is asserted, the data transfer between the serial-to-parallel register and the internal register is delayed by one CLK period. When the transfer of the data is delayed by one CLK period, one extra bit is shifted into the serial-to-parallel register and one bit is lost at the other end. When the data is then captured in the internal register, it appears as if the data is shifted by one bit.

Figure 3 shows the same behavior as Figure 2 except that a Bitslip operation is executed when the second byte is captured in the serial-to-parallel register. Data is not transferred to the parallel register after capturing eighth bits, but after nine bits. It appears as if the pattern is shifted by one bit. The first bit shifted in is lost at the bottom (end) of the serial-to-parallel register.

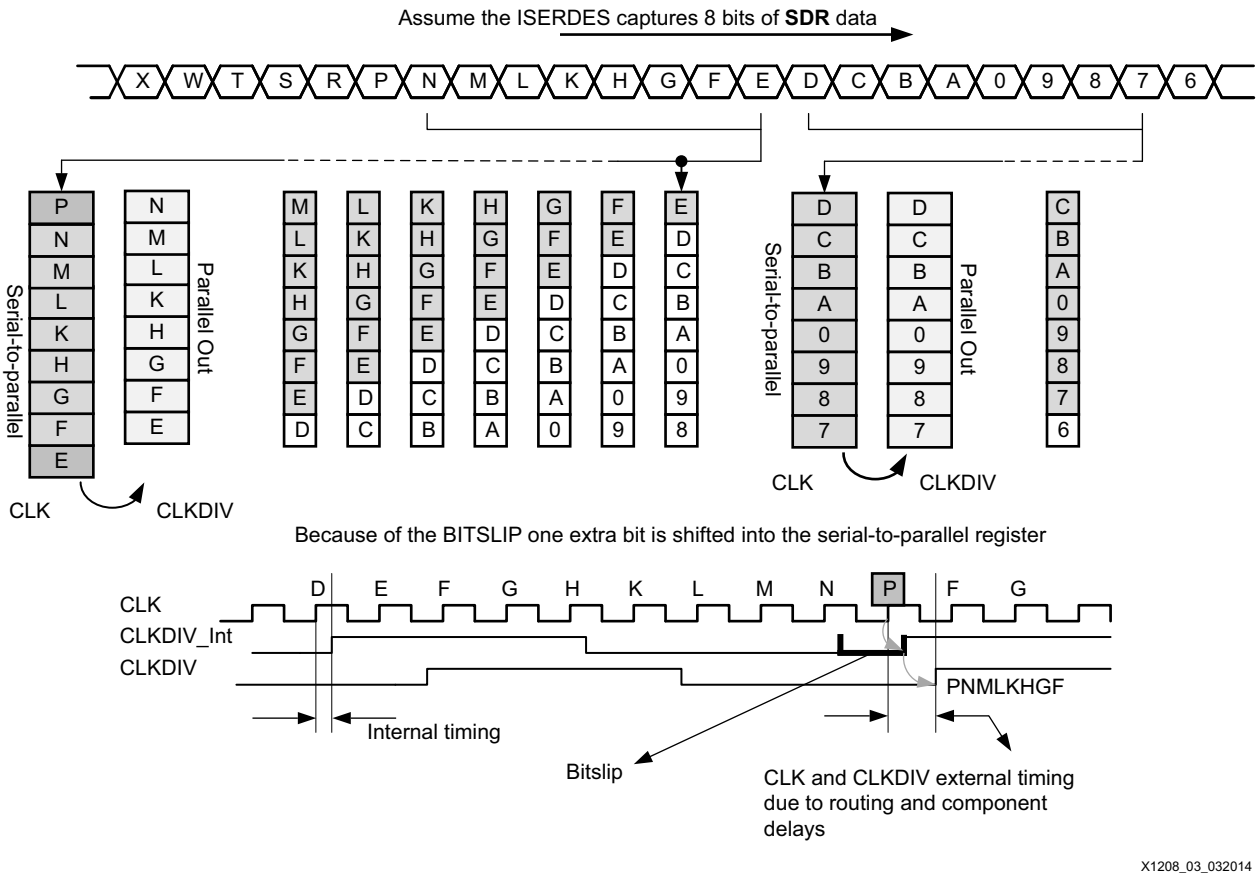


Figure 3: **Capturing Bits Using Bitslip**

Each time the BITSLIP input is held high for half a CLKDIV period, a Bitslip operation is performed on the data. Contiguously capturing the same input bit pattern results in the bits being shifted around. This mechanism can thus easily be used to align data to a pattern or align one data channel to another.

Bitslip in Logic Functional Description

The Bitslip function can also be implemented using general interconnect at the output of the I/O logic. This means that the function must be performed on the parallel deserialized data instead of, as explained in Native Bitslip Function in Previous Architectures, on the incoming serial data.

A circuit is provided that makes it possible to Bitslip or shift bits in parallel data words at CLKDIV rate. Several possible circuits can be constructed to perform this action. A dynamic shift and a predetermined shift solution are now described.

Solution A

Rotate bits in serially ordered parallel registers at CLKDIV rate. In order to Bitslip or shift through all possibilities, it will take up to as many CLKDIV clock cycles as the width of a register (see Figure 4).

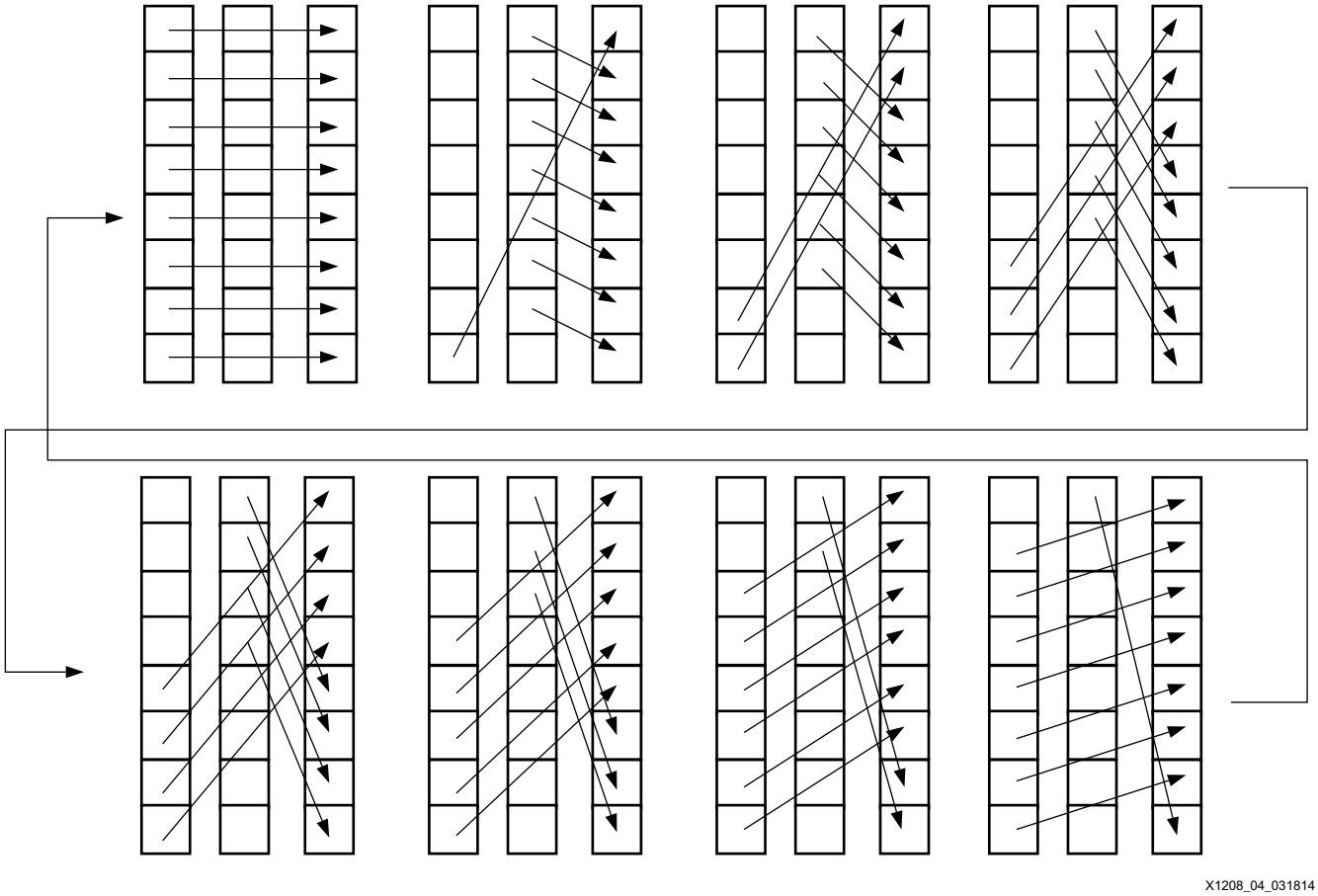


Figure 4: **Serial-Oriented Bitslip in Logic**

All shift or Bitslip steps possible on an 8-bit word are shown in Figure 4. To perform these actions, a minimum of two 8-bit registers and a multiplexer are needed. Figure 5 to Figure 7 show in detail what happens when a single Bitslip step is performed.

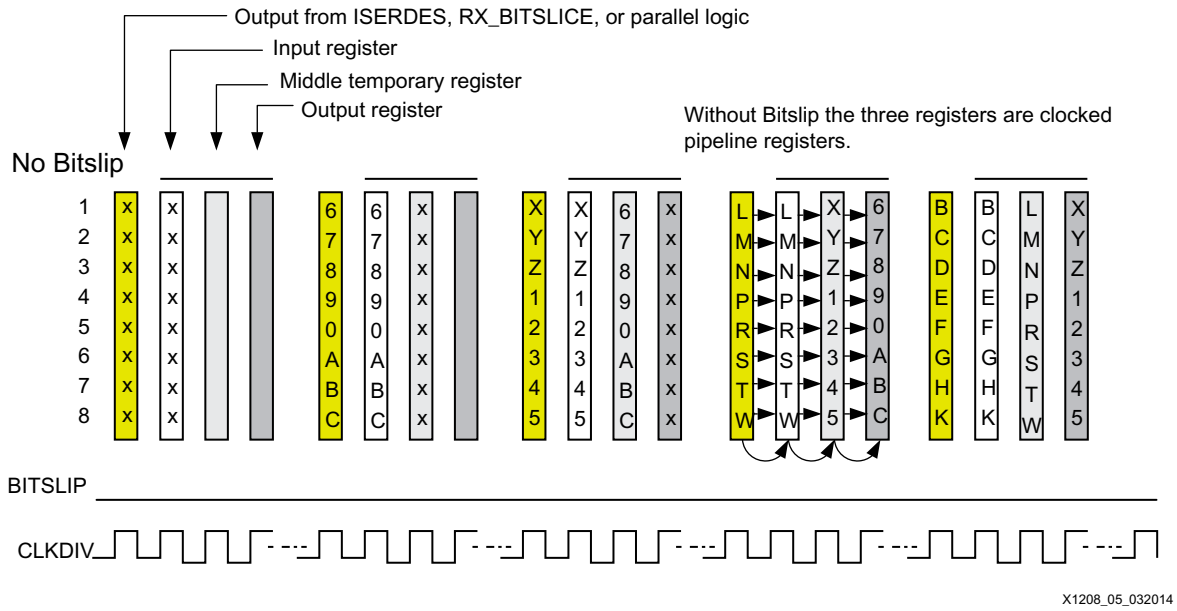


Figure 5: **Detailed Bitslip Operation - Step a**

- Step a: At the beginning (Figure 5), and without having any Bitslip performed, the registers act as pipeline registers. Parallel data is just passed through the registers at CLKDIV clock rate.

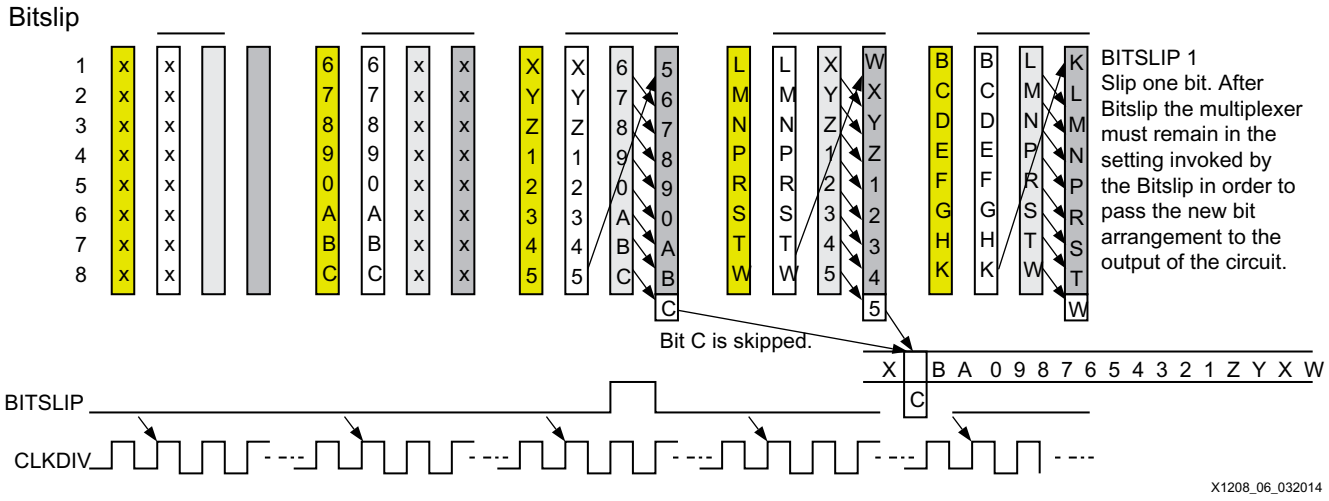


Figure 6: **Detailed Bitslip Operation - Step b**

- Step b: Figure 6 shows what happens when a Bitslip is invoked. As in the previous device operating modes, one bit is skipped and all other bits are rearranged to a new setting. This rearrangement of the bits must be maintained after the Bitslip operation in order to reflect the Bitslip operation in further operation of the design.

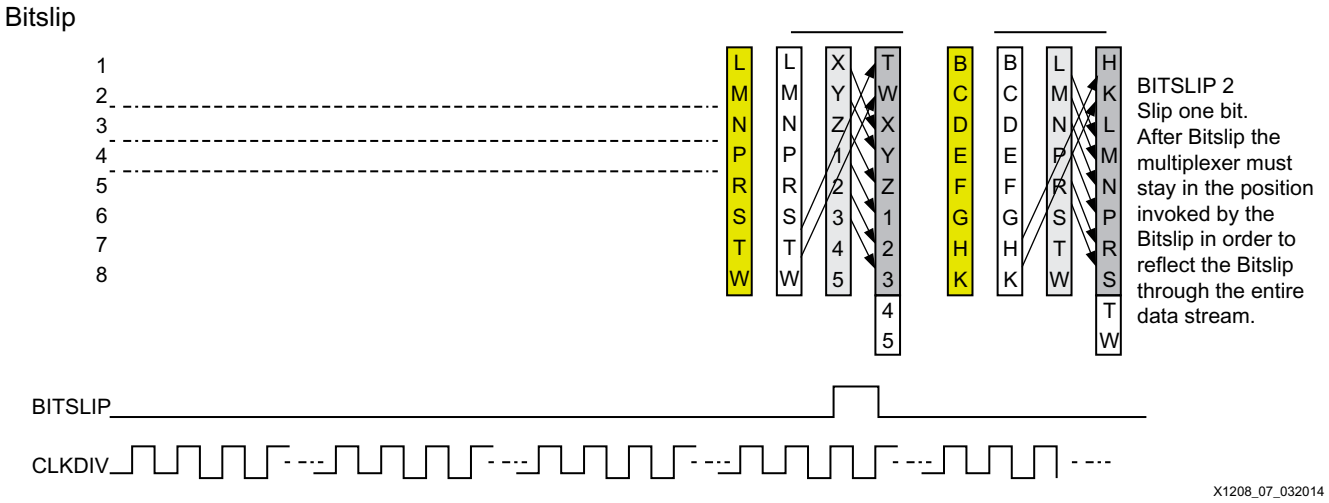


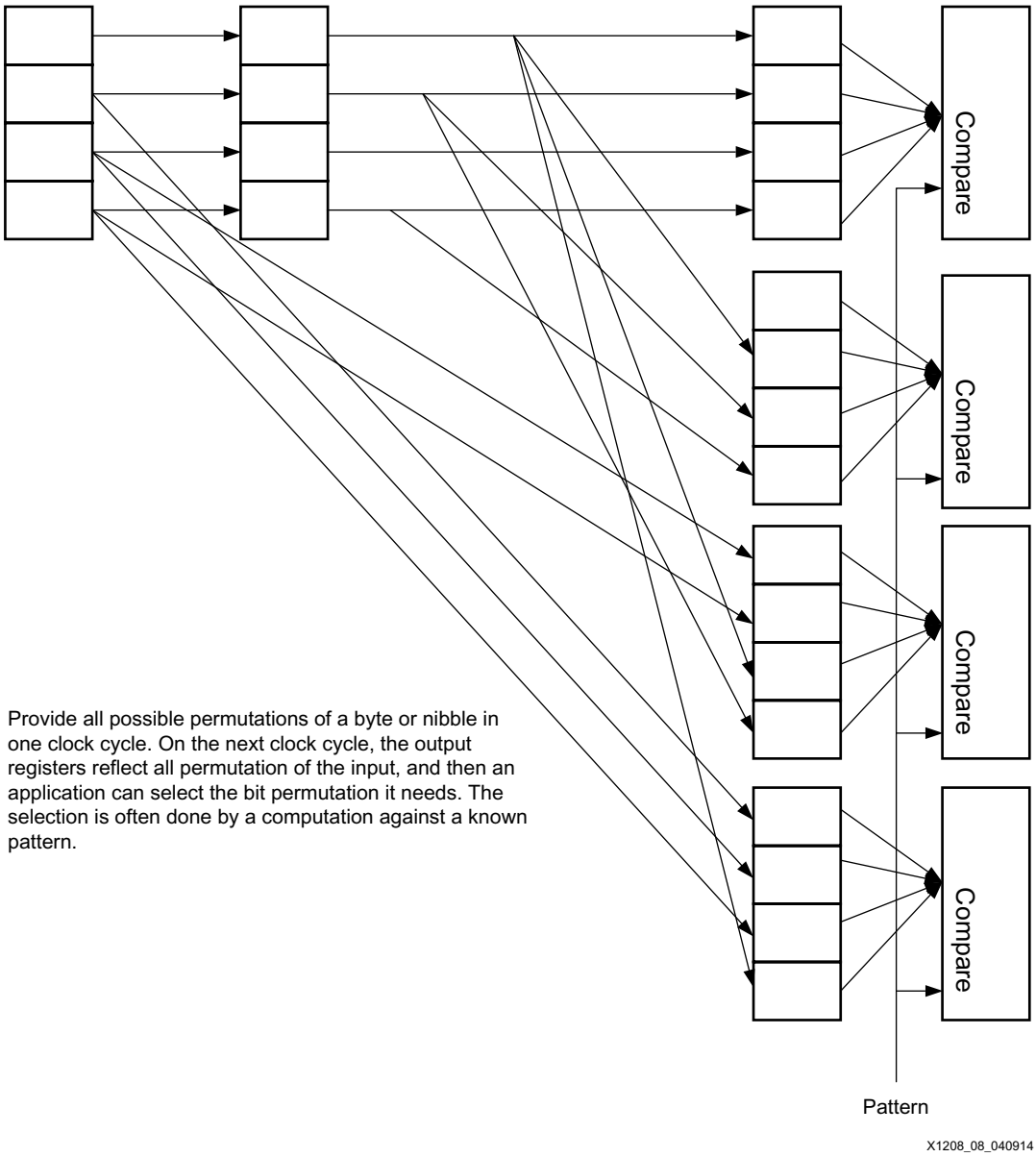
Figure 7: **Detailed Bitslip Operation - Step c**

- Step c: Figure 7 shows a new Bitslip in which two bits are skipped after the previous Bitslip operation. For the data, this looks exactly as in the serial modes of the previous device families, i.e., as if all bits are shifted by two bits.

The registers labeled in Figure 5 as *Output from ISERDES, RX_BITSLICE, or parallel logic* and *Input register* can be combined in one single register. Combining the registers saves one CLKDIV clock pipeline stage, while using the registers separately can provide better timing performance. The option to do this is provided as a generic attribute in the reference design.

Solution B

Take the output of an ISERDES, RX_BITSLICE, or other parallel logic and rotate all bits simultaneously into a set of parallel placed registers. This solution takes one CLKDIV cycle to shift through all possible values. At the start of operation, it takes two clock cycles to obtain a valid output: an initial clock cycle to load a history register and a second clock cycle to load all bit permutations in the output registers. From then on, each clock cycle results in valid output data. An application can then select the nibble or byte with the desired bit permutation. This solution is ideal when a pre-determined pattern must be detected and matched (see Figure 8).



Provide all possible permutations of a byte or nibble in one clock cycle. On the next clock cycle, the output registers reflect all permutation of the input, and then an application can select the bit permutation it needs. The selection is often done by a computation against a known pattern.

Figure 8: **Four Bit-Wide Parallel-Oriented Bitslip in Logic**

Bitslip in Logic Design Description

The reference design contains both described circuits, and the chosen options determine what circuit and what options are used. The reference design comes as a single instantiable component with a set of selectable options. Each of the two possible circuits uses its own set of options to apply design features.

The reference design component is shown in Figure 1. The pin functionality is given below:

- DataIn: 4- or 8-bit input. In 4-bit mode, use the LSB nibble.
- BitSlip: Control input that when pulsed High, invokes a Bitslip for solution A and a circuit enable for solution B.
- SlipVal: 3-bit input to let the circuit Bitslip a defined amount of times before enabling the output.
- CompVal: 8-bit input used to detect a pattern in the data. In 4-bit mode, use the LSB nibble.

- Ena: Circuit enable, active High.
- Rst: Circuit reset, active high.
- Clk: Clock, normally equal to the CLKDIV clock.
- DataOut: 4- or 8-bit output. In 4-bit mode, use the LSB nibble.
- ErrOut: Error and status output:
  - “Slip” mode: Pulses High when eight Bitslips are performed.
  - “Nmbr” mode: Indicates that output data is ready after “Nmbr” of Bitslips.
  - “Comp” mode: Indication that the requested pattern is detected.

Solution A

The customization options using source code generics are:

- C_DataWidth: This sets the data width of the design. Allowable options for this parameter are 4 and 8. Eight is the default value and expected to be the most useful because it provides the best use of the I/O logic and lowest internal clock rate (easiest to obtain timing goals).
- C_InputReg: When this option is set to 1, an extra input register is present at the input of the circuit. By default, this option is turned off by setting to 0. An extra register adds an extra clock cycle.
- C_Function: Three options out of four apply to this part of the design:
  - Slip: This is Bitslip as available in previous device families, and Bitslip per bit when the Bitslip input is pulsed High for one clock cycle. When the Bitslip input is held High for several clock cycles, the effect is the same as when Bitslip is pulsed for one clock cycle.
  - Nmbr: This option Bitslips the input data for the value given at the SlipVal input. The output is provided when all requested Bitslips are done. The Bitslip input serves as a load for the value applied at the SlipVal inputs. Pulse the Bitslip input High for one clock cycle.
- Comp: This option Bitslips the input data until the pattern given at the CompVal inputs has been detected, then the output is enabled.

Figure 9 shows the implementation of solution A.

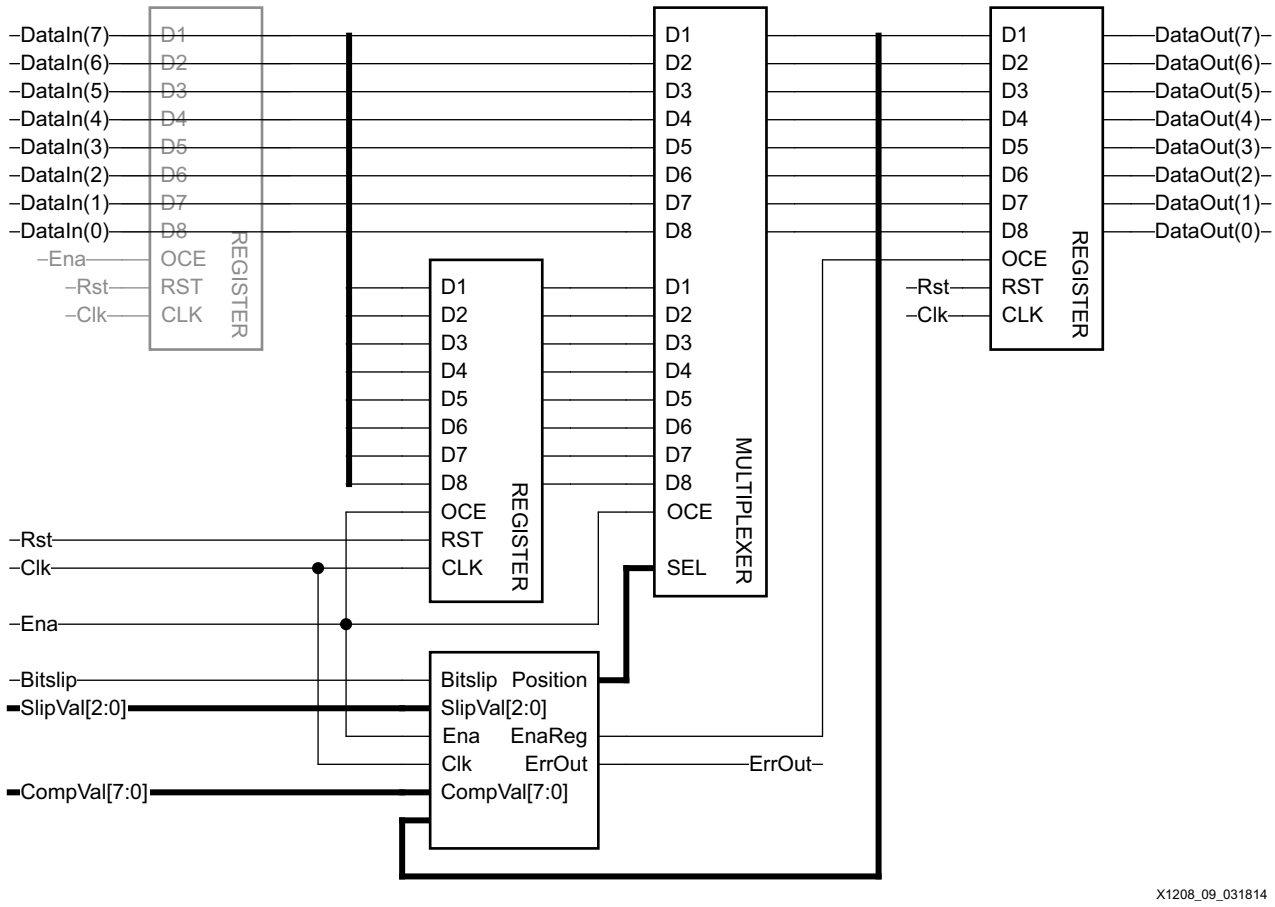


Figure 9: **Implementation of Solution A**

Solution B

The customization options using source code generics are:

- C_DataWidth: This sets the data width of the design. Allowable options for this parameter are 4 and 8. Eight is the default value and expected to be the most useful, because it provides the best use of the I/O logic and lowest internal clock rate (easiest to obtain timing goals).
- C_InputReg: When this option is set to 1, an extra input register is present at the input of the circuit. By default, this option is turned off by setting to 0. An extra register adds an extra clock cycle.
- C_Function: The fourth option applies to this design solution:
  - FstC: Fast compare. The Bitslip inputs initiate a compare for a given pattern at the CompVal inputs. Data is output when the provided pattern is detected.

Figure 10 shows the implementation of solution B.

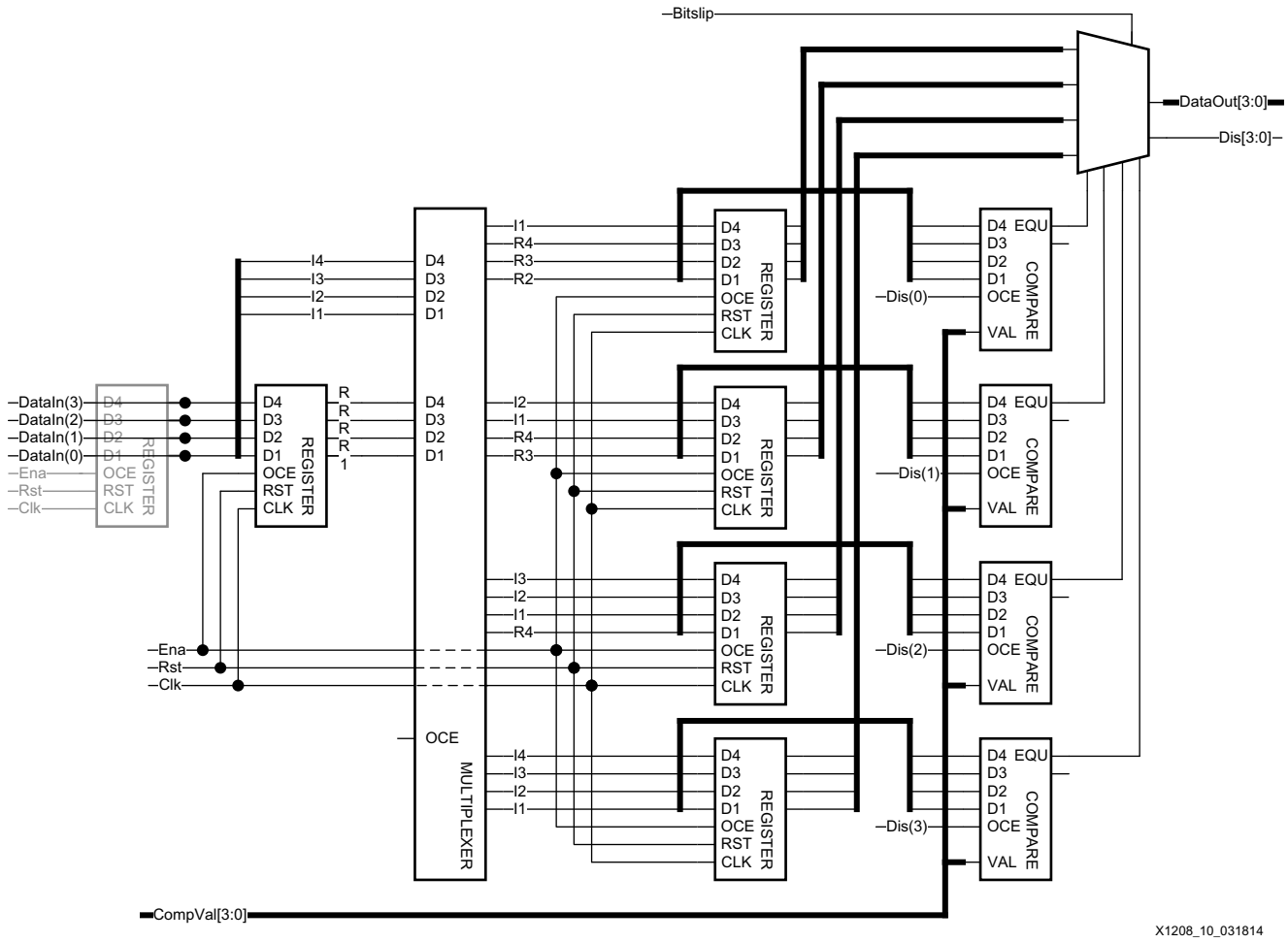


Figure 10: **Implementation of Solution B**

An implementation constraints XDC file places the reference design in squared boundaries. However, when the design is used in an application, you can adapt this XDC file to the needs of the application. Figure 11 and Figure 12 show the implemented design in “Slip” mode for solution A and “FstC” mode for solution B.

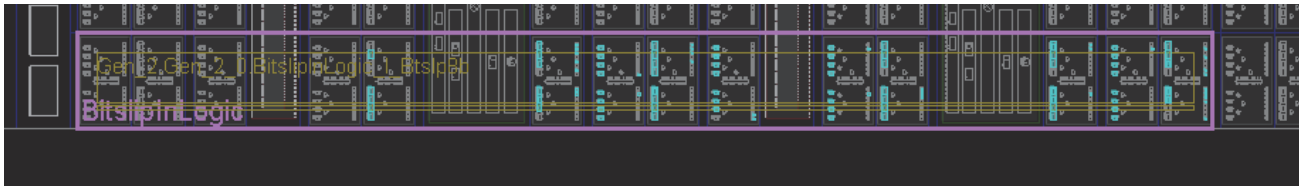


Figure 11: **Implemented Design in “Slip” Mode**

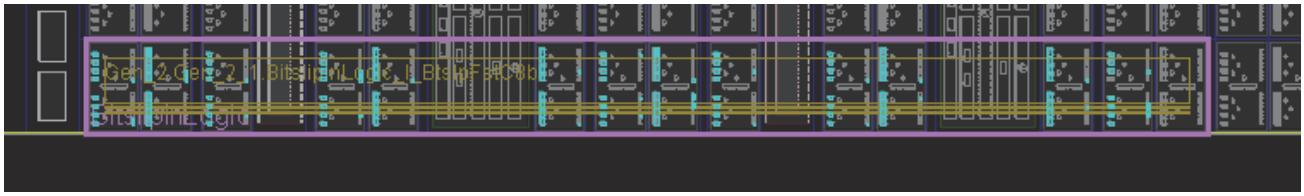


Figure 12: **Implemented Design in "FstC" Mode**

Reference Design

The reference design files for this application note can be downloaded from:

https://secure.xilinx.com/webreg/clickthrough.do?cid=356522

The reference design matrix is shown in Table 1.

Table 1: **Reference Design Matrix**

Parameter	Description
General	
Developer name	Marc Defossez
Target devices (stepping level, ES, production, speed grades)	UltraScale devices
Source code provided	Yes
Source code format	VHDL
Design uses code and IP from existing Xilinx application note and reference designs, CORE Generator software, or third party	No
Simulation	
Functional simulation performed	Yes
Timing simulation performed	Yes
Test bench used for functional and timing simulations	Yes
Test bench format	VHDL
Simulator software/version used	Mentor Graphics Questa Advanced Simulator 10.2a
SPICE/IBIS simulations	No
Implementation	
Synthesis software tools/version used	Vivado Design Suite 2014.1
Implementation software tools/versions used	Vivado Design Suite 2014.1
Static timing analysis performed	Yes
Hardware Verification	
Hardware verified	Yes
Hardware platform used for verification	KCU105 board

Table 2 summarizes the reference design utilization.

Table 2: **Reference Design Utilization Summary**

Component	Percentage (%)	Total Number	Number Utilized
CLB	0.04	30,300	14
LUT as logic	0.04	242,400	98
LUT as flip-flop pairs	0.04	242,400	101

Notes:

1. Design implemented using Vivado_2014.1 rev: EA851449 (64-bit).
2. UltraScale device used: XCKU040-FFVA1156-2.
3. Options of the design:
   - C_DataWidth: 8
   - C_InputReg: 1
   - C_Function: FstC
4. Synthesis option: -mode out_of_context.
5. Implementation option: retarget.

Figure 13 shows the reference design directory setup.

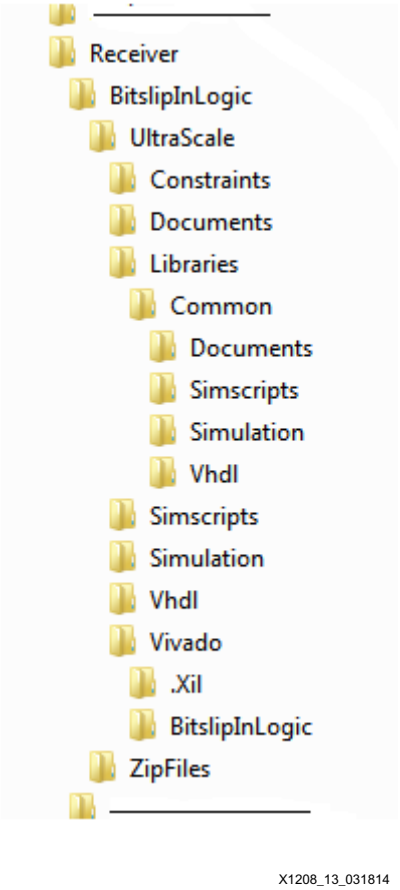


Figure 13: **Reference Design Directory Setup**

Conclusion

This Bitslip reference design described in this application note performs the same functionality as the native Bitslip functionality embedded in the ISERDES of 7 series and Virtex-6 FPGAs. This design offers extra options not available in the original 7 series and Virtex-6 FPGA ISERDES solutions. The general interconnect must be used when the functionality available in this design is needed in a 7 series or Virtex-6 FPGA design. Therefore, this Bitslip reference design meets the requirements and goals of Bitslip offered in previous device families.

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
05/16/2014	1.0	Initial Xilinx release.

Notice of Disclaimer

Automotive Applications Disclaimer