XAPP690 (v1.0) October 6, 2003

Using Block SelectRAM Memories as Serializers or Deserializers

Author: Marc Defossez, Nick Sawyer

Summary

This application note describes how block memories efficiently can implement a serializer or a deserializer function or both with or without pattern matching capabilities in the Virtex™-II, Virtex-II Pro™, and Spartan™-3 architectures. The used method can deserialize or serialize with or without data storage. The resulting implementation can be used in either dual-port or single-port mode, with full functionality at the maximum speed of the Block SelectRAM.

Introduction

Since the original Virtex device was introduced, users have had access to block memories in the Xilinx architecture. These 4 Kbit blocks in the Virtex, VirtexE, and Spartan-II devices were increased in size to 18 Kbit blocks in the Virtex-II, Virtex-II Pro, and Spartan-3 devices. These blocks are fully synchronous, true dual-port structures. That is, the user can read from or write to each port independently (with the exception of simultaneous reads and writes to the same address). In addition, each port has a separate clock, and the data widths for each port are independently programmable. Figure 1 shows a block diagram of the dual-port RAM blocks.

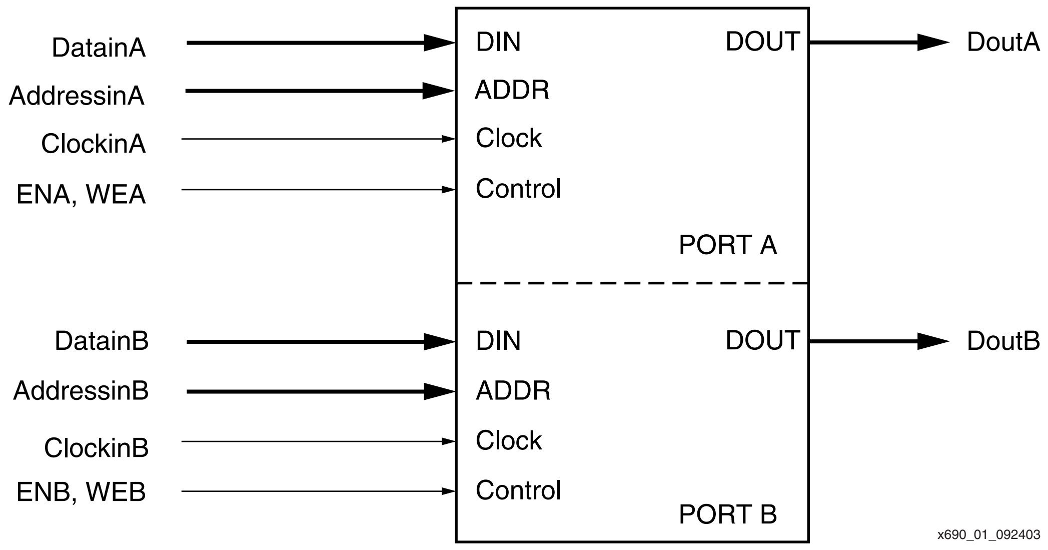

Figure 1: **Basic Block RAM Structure**

Today's high-speed communications between chips, boards, or systems typically are done serially. Xilinx has implemented dedicated Multi Gigabit Transceiver (MGT) devices in each Virtex-II Pro FPGA. Alternatively, other FPGAs can use externally placed MGTs for high-speed serial communication designs.

Often, however, serial communication at lower speeds is needed. Then any device with Block SelectRAM can implement serializers/deserializers using the ideas described in this application note.

The RAM-based serial-to-parallel and parallel-to-serial converters from this application note can run at the maximum clock speed of the Block SelectRAM. With some extra logic, clock-data recovery, pattern recognition, Block SelectRAM storage, and a full-featured ''Multi Megabit Transceiver" can be implemented.

Deserialization

By definition, deserializing means one bit of data is read at a given clock rate into a device, and after a predetermined number of clock ticks, the serial shifted data is obtained at a parallel output port. Figure 2 shows a generic deserializer.

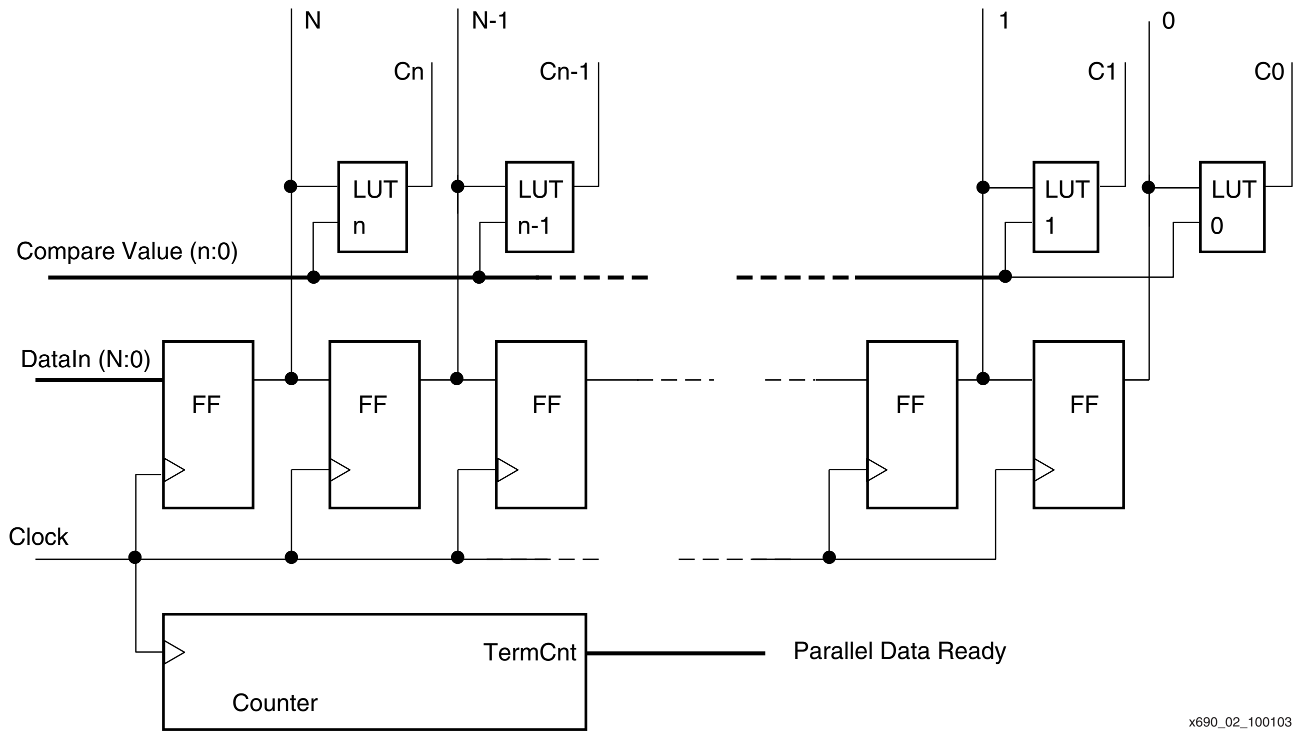

Figure 2: **Generic Deserializer**

In most cases, the device that stores the serial incoming bits is a register constructed of regular flip-flops, and the number of obtained parallel bits is determined by a (preset) counter.

Often, a given data pattern must be recognized to be valid, for instance a framing application, before data can be taken from the parallel port. This validation is done with a comparator built from logic gates.

FPGA deserializer implementations are based on the following principles: slice flip-flops are used to store the data bits, and LUTs are used to build the comparison logic.

Single-Port Deserialization Implementation

Fixed Address

Figure 3 shows the single-port deserializer circuit with fixed block RAM address. The corresponding timing diagram is shown in Figure 4.

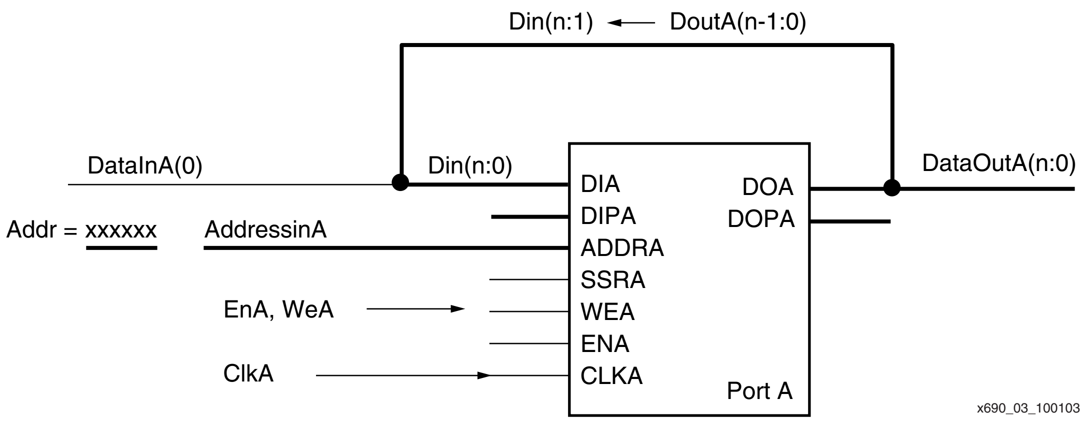

Figure 3: **Single-Port Deserializer Circuit**

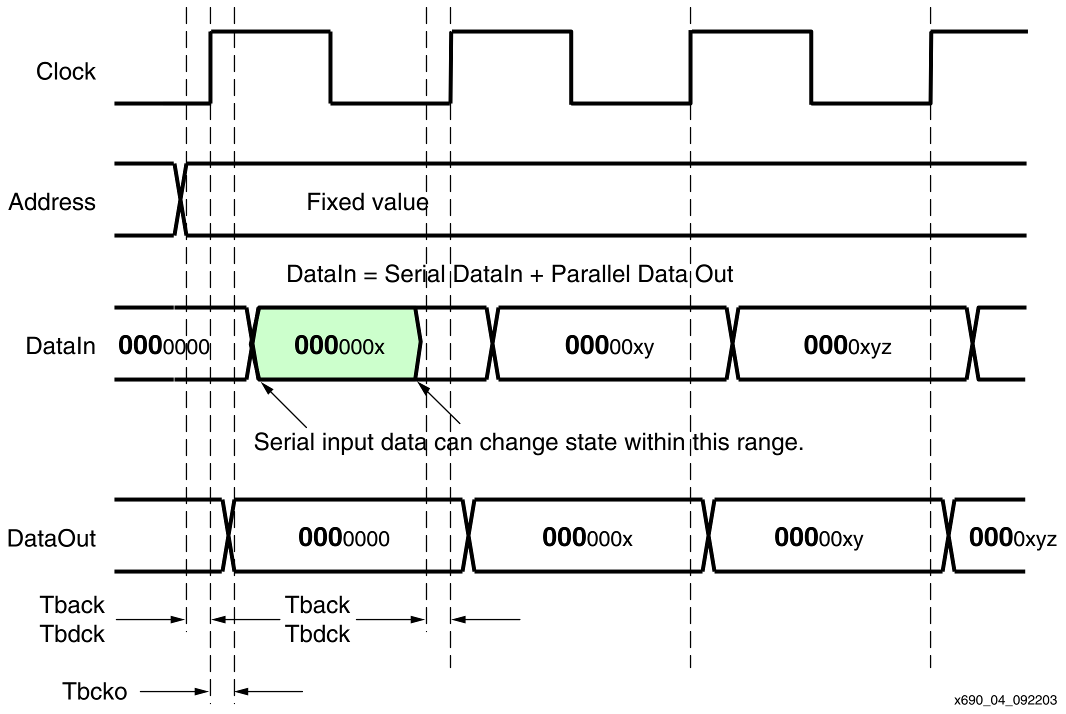

Figure 4: **Single-Port Deserializer Timing Diagram**

This design functions due to the latency of the Block SelectRAM. Data appears at the output of the Block SelectRAM in write-through mode after a delay, Tbcko, following the write clock edge.

The input of the Block SelectRAM is composed of serial input data and one shifted bit of feedback data from the RAM output (see Table 1).

Table 1: **Single-Port Deserializer I/Os**

DataIn	**BRAM**		**DataOut**
Serial In	DIA 0	DOA 0	Bit 0
Bit 0	DIA 1	DOA 1	Bit 1
Bit 1	DIA 2	DOA 2	Bit 2
...	...	...	...
Bit n-2	DIA n-1	DOA n-1	Bit n-1
Bit n-1	DIA n	DOA n	Bit n

The data loopback routing in the FPGA must be achieved in a time less than the input clock period minus the data setup time [(Tpwh + Tpwl) – Tbdck]. Timing constraints are necessary.

Data is written into the RAM array on the rising clock edge. This data appears at the output of the RAM after a delay, Tbcko. The new RAM array output is equal to the serial input data concatenated with the previous output of the Block SelectRAM.

A number of clock cycles later, depending on the desired SERDES factor, the serial input pattern will have been converted to parallel. The maximum speed obtained for the circuit is the maximum clocking speed of the Block SelectRAM (Virtex-II Pro FPGA, –6 = 380 MHz; Spartan-3 FPGA, -4 = 250 MHz). The parallel converted data is available at clock speed / number of bits.

The width of the parallel output data depends on the chosen Block SelectRAM data output width. For instance, if the required SERDES factor is 8, then a data width of 8 is chosen. In this design case, the depth of the Block SelectRAM does not apply because the address is fixed to one particular location (address 0xFFFF is straightforward, requiring no routing).

If the parallel output is not captured at the correct moment, then the parallel data continues to shift along with the serial incoming data. This feature can be used only to capture data when a dedicated pattern is recognized.

Variable Address

It is possible to use the RAM array simultaneously as a deserializer and as storage for parallel converted data. When the fixed address input value mentioned above is replaced by a counter that is incremented (decremented) when the maximum data width is reached, it is possible to store the parallel converted values in memory. The address counter changes state at a clock/data-width rate, however, the address must be ready at the memory address inputs in a period less than the clock period minus the address setup time [(Tpwh + Tpwl) – Tback].

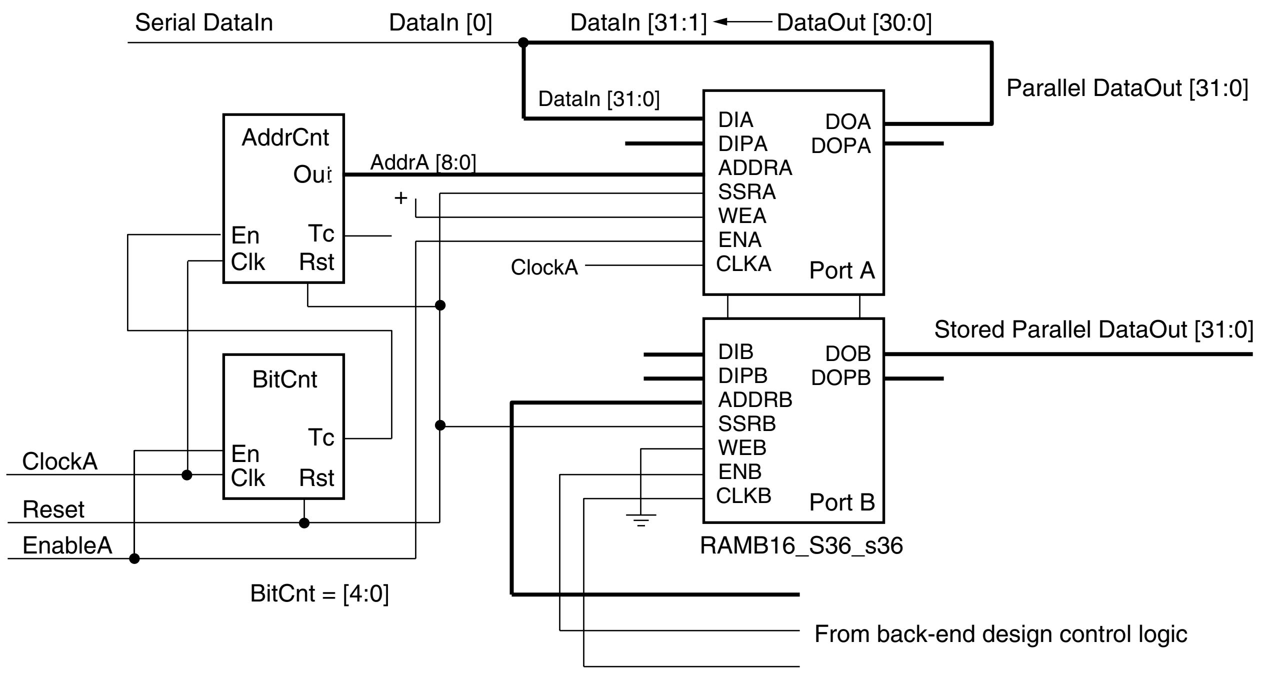

Figure 5: **Single-Port Deserializer with Second Port Readout Functionality**

Using this technique it is possible to build a circuit as shown in Figure 5. A small counter equal to the width of the memory data output that is running at the serial bitstream clock rate determines when the address counter is incremented or decremented.

Serial-to-parallel converted data can be captured at the terminal count of the bit counter, at the moment when the address is changed. Alternatively, port B of the Block SelectRAM can be used to read the stored, serial to-parallel converted data into a back-end design. The same precaution as when reading FIFO-type memories must be taken. That is, do not read the address used by the serial-to-parallel conversion.

The next step with these designs is to store data either only in memory when a pattern is matched or from the point a certain data pattern was matched. Figure 6 shows a design where data is stored when a pattern is matched. This pattern also can be a variable value depending, for instance, on the state of the design at a certain point in time. In the design shown in Figure 6, some extra logic has been added to indicate that no pattern was found (missed) after a number of trials.

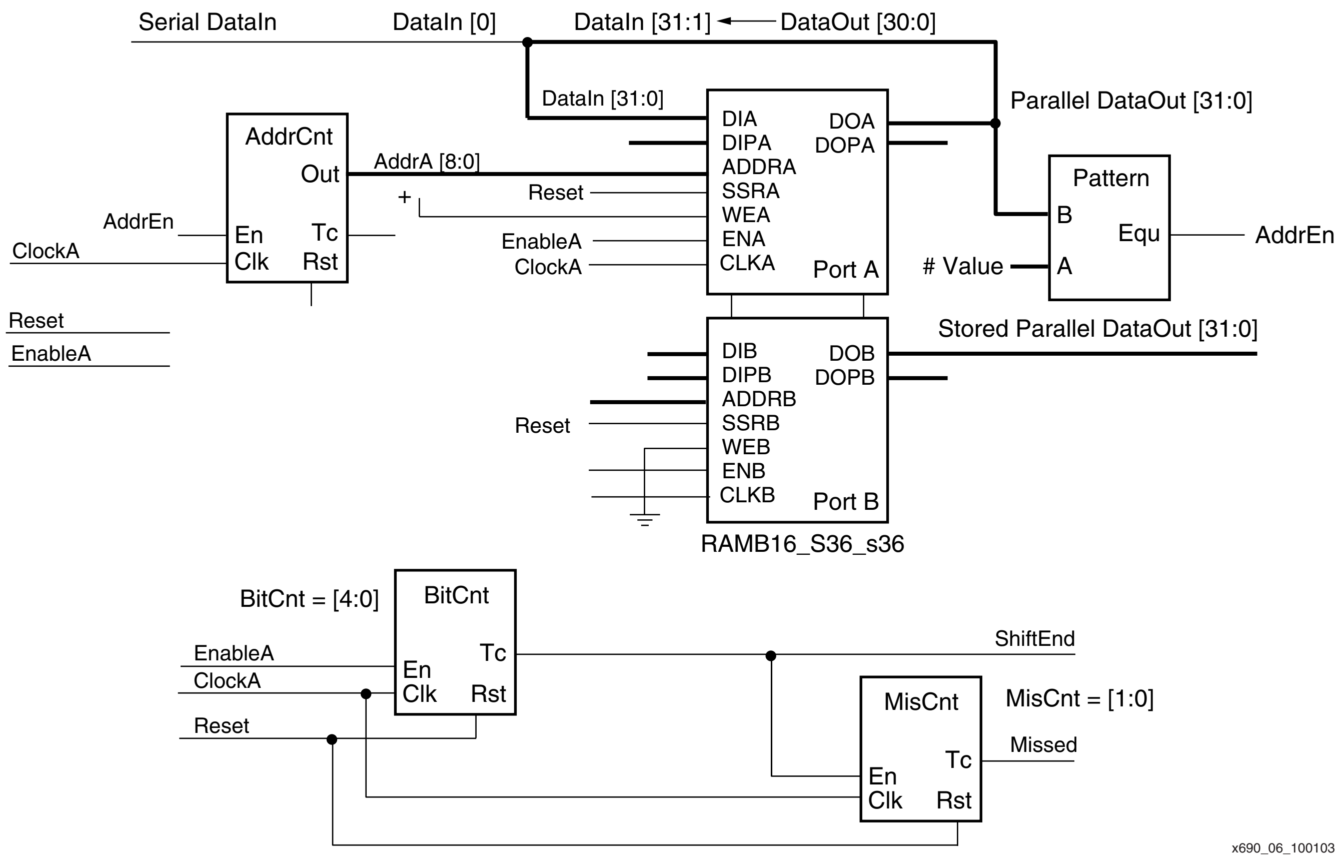

Figure 6: **Single-Port Deserializer with Second Port Readout and Pattern Matching**

Dual-Port Deserialization Implementation

The dual-port implementation of the serial-to-parallel converter allows each block RAM to implement two serial-to-parallel converters, and is limited to the fixed address implementation. Only one address place per Block SelectRAM port can be used, and there is no data storage ability. Each port of the RAM uses a different address. For the functionality description, read the single-port implementation described in "Single-Port Deserialization Implementation," and duplicate it for both RAM ports.

Figure 7 shows the circuit for the dual-port deserialization implementation.

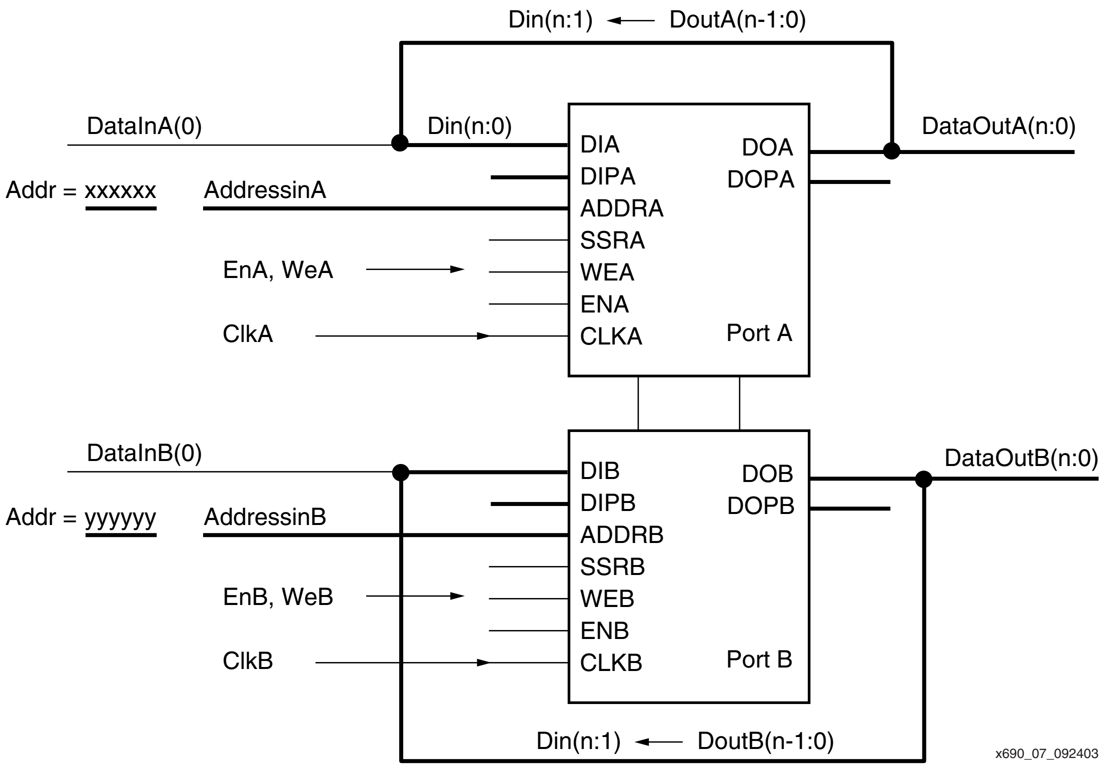

Figure 7: **Dual-Port Deserializer Functionality**

Incoming DDR data can be deserialized directly using a dual-port implementation as shown in Figure 8. The same serial input data is connected to both RAM input ports. Odd bits are clocked in on the rising clock edge into port A, and even bits are clocked in on the falling clock edge into port B. Both ports then are read at the same time interleaving the obtained data. This way it is possible to double the serializer speed.

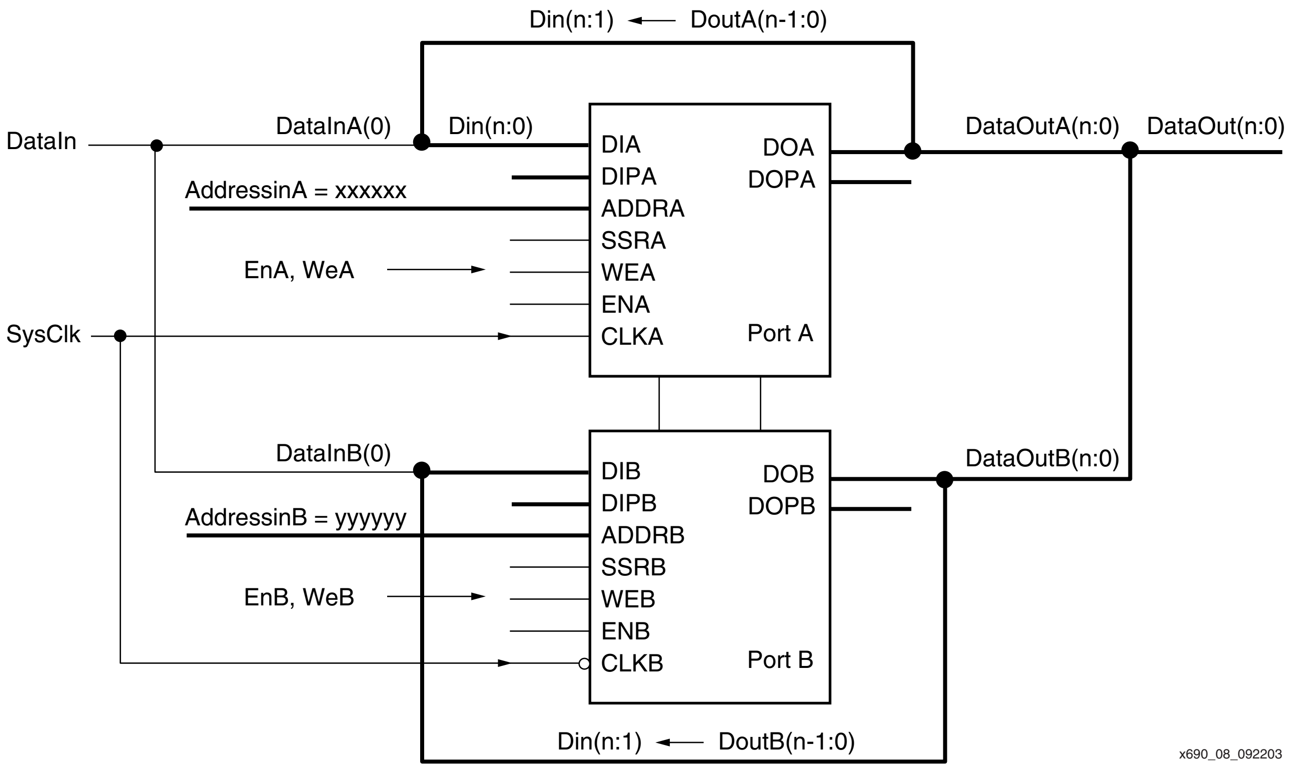

Figure 8: **Dual-Port Deserializer with Interleaved DDR**

Serialization

By definition, serializing means *n* number of bits are stored in a register at the rate of a load enable signal and the bits are shifted serially at a given clock rate out at the MSB or LSB position of the register. The parallel loading of the register happens once every *n* clock bits and usually is controlled through the terminal count of a bit counter.

In most cases, the device used to store the parallel-loaded bits is a register constructed of regular flip-flops preceded by a small logic function enabling the load or shift operation. This function is performed in a LUT.

Figure 9 shows a generic serializer circuit.

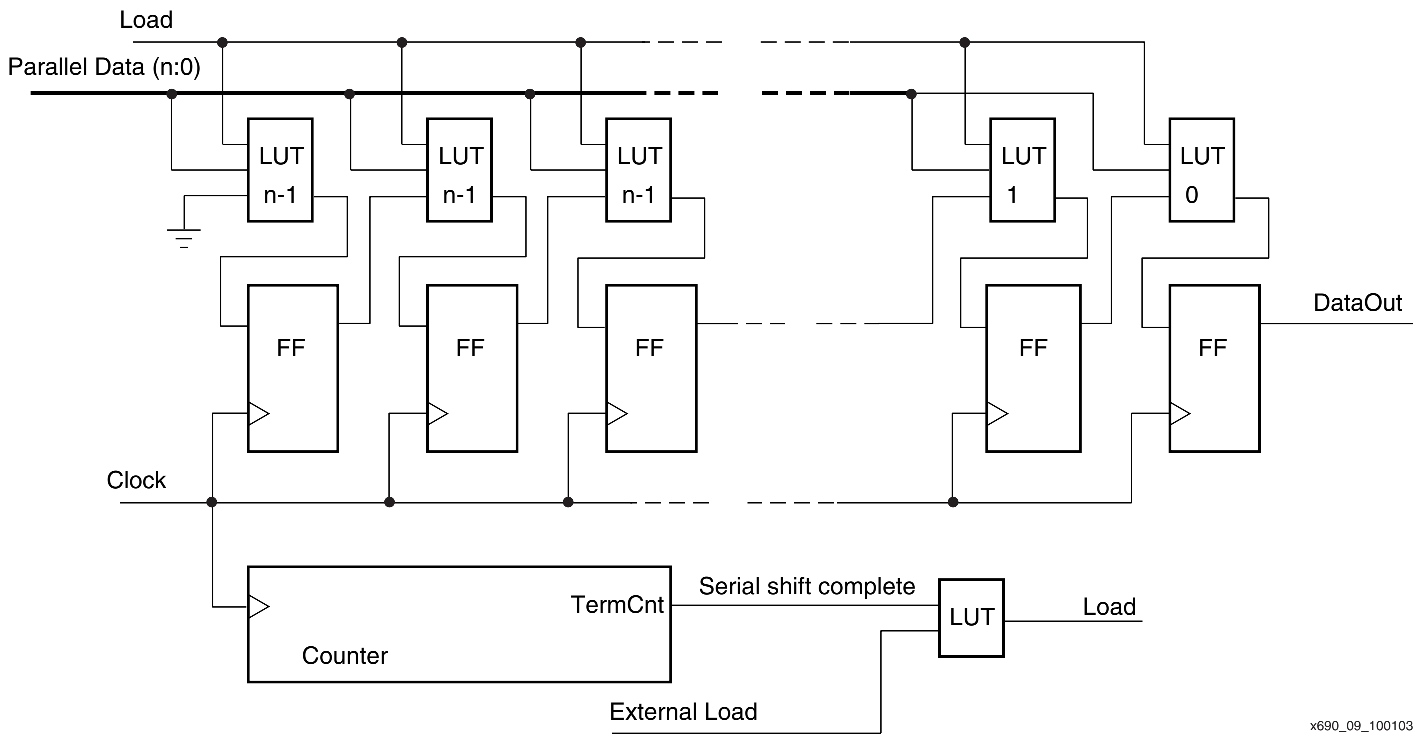

Figure 9: **Generic Serializer**

FPGA serialization implementations are based on these principles: slice flip-flops and LUTs are used to store, load, and shift the data bits.

Single-Port Serialization Implementation

Fixed Address

The operation of the single-port circuit with fixed address is shown in Figure 10. The corresponding timing diagram is shown in Figure 11.

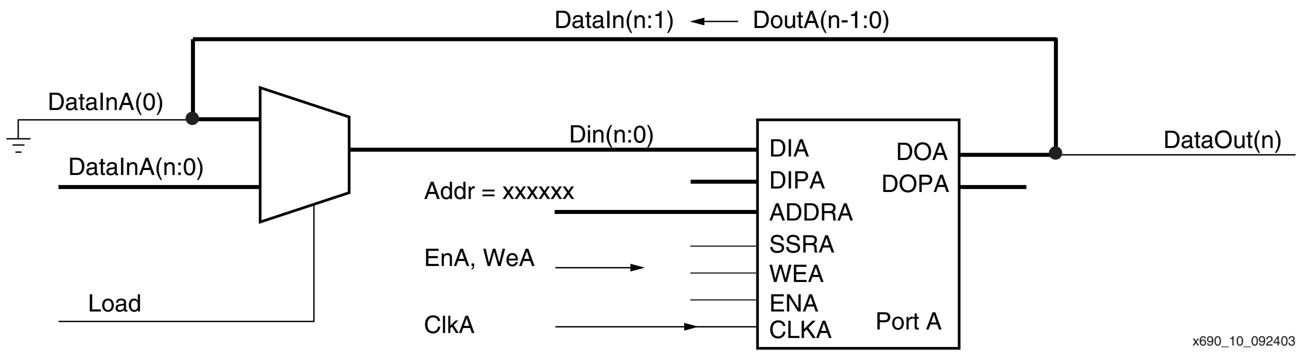

Figure 10: **Single-Port Serializer**

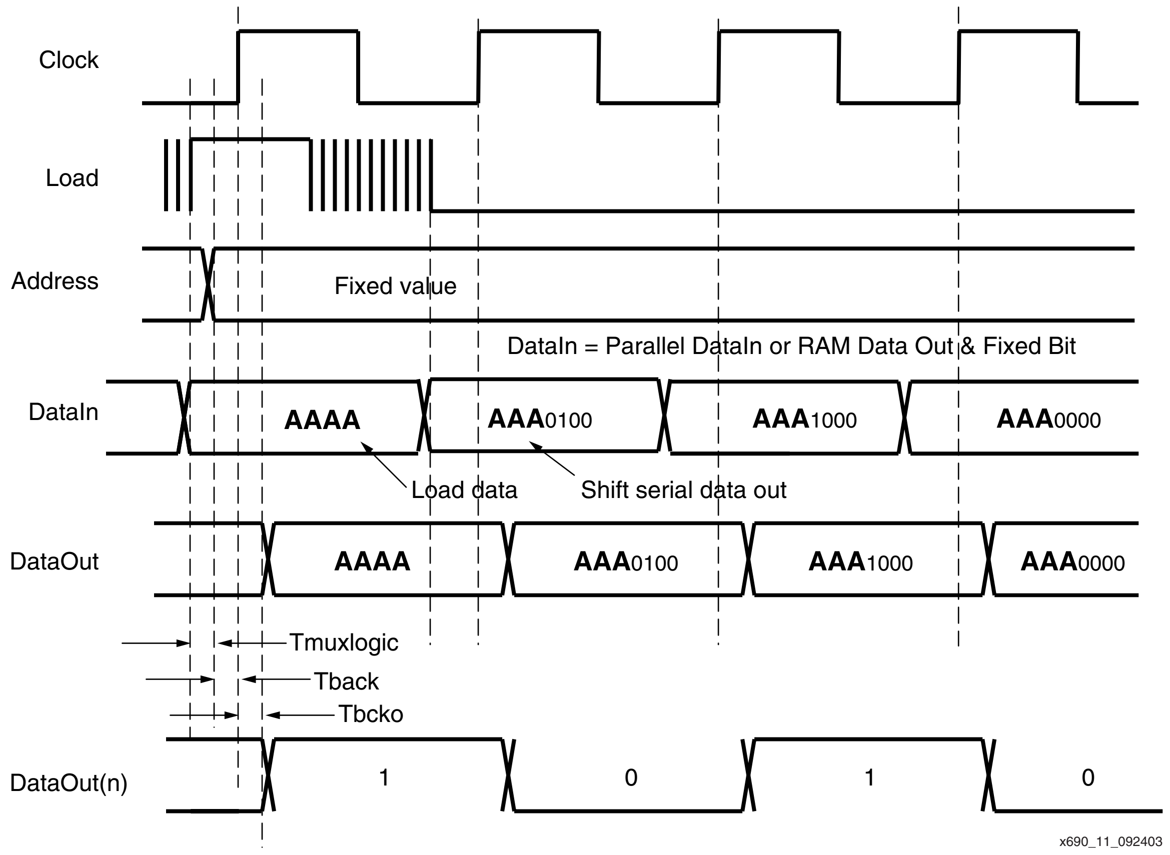

Figure 11: **Single-Port Serializer Timing Diagram**

This design functions due to the latency of the Block SelectRAM. Data appears at the output of the Block SelectRAM in write-through mode after a delay, Tbcko, following the write clock edge.

The input of the Block SelectRAM is composed of the output of a multiplexer selecting at the rate of a load strobe, parallel input data, or the feedback data of the RAM output, shifted one bit with MSB or LSB fixed Low or High (see Table 2).

Table 2: **Dual-Port Deserializer I/Os**

DataIn Load		**BRAM**		**DataOut**
1	**0**			
Bit 0	Gnd	DIA 0	DOA 0	Bit 0
Bit 1	Bit 0	DIA 1	DOA 1	Bit 1
Bit 2	Bit 1	DIA 2	DOA 2	Bit 2
...	...	...	...	...
Bit n-2	Bit n-3	DIA n-1	DOA n-1	Bit n-1
Bit n-1	Bit n-2	DIA n	DOA n	Serial Out

The data loopback routing in the FPGA must be achieved in a time less than the clock period minus the data setup time [(Tpwh + Tpwl) – Tbdck] minus the multiplexer logic delay time (Tmuxlogic). Timing constraints are necessary.

At the clock edge, when Load is High, parallel data is written into the RAM array from an external source. This data appears at the output of the RAM after a delay, Tbcko. The new RAM array output is equal to the parallel input data. When Load is Low, the output of the Block SelectRAM (shifted one bit and most-significant bit or least-significant bit made Low or High) is fed back to the input of the RAM through a multiplexer. After a delay, Tbcko, the RAM output shows the new value. The Msb or Lsb output of the Block SelectRAM shows a serial bit pattern that represents, after a number of clock cycles, the parallel input pattern.

The maximum speed obtained for the circuit is equal to the multiplexer logic delay + some routing delay + the clock speed of the Block SelectRAM (Virtex-II Pro FPGA, –6 > 300 MHz SDR; Spartan-3 FPGA, -4 = 200 MHz SDR).

The number of bits in the serial output depends of the chosen Block SelectRAM data width. In this case, the depth of the Block SelectRAM is not applicable because the address is fixed to one particular place (use address 0x0000 or 0xFFFF, which is easy and requires no routing).

When the parallel output is not loaded at the correct moment, the serial data continues to shift and displays all zeros or ones, depending on the fixed input bit.

Variable Address

It is possible to use the RAM array simultaneously as storage for data and serialized data using both ports of the Block SelectRAM. When the fixed address input value is replaced by a counter, which is incremented (decremented) when all bits of a data word are shifted, it is possible to work with data stored in the memory. The address counter changes state at the clock rate divided by the number of data bits. The address data must be ready at the memory address inputs in a period less than the clock period minus the address setup time [(Tpwh + Tpwl) – Tback] and multiplexer delay (Tmuxlogic).

Using this technique, it is possible to build a circuit as shown in Figure 12. Port A of the Block SelectRAM is used as normal memory, with the possibility of storing *n* data words. Port B serves as a parallel-to-serial converter on data stored in the RAM array.

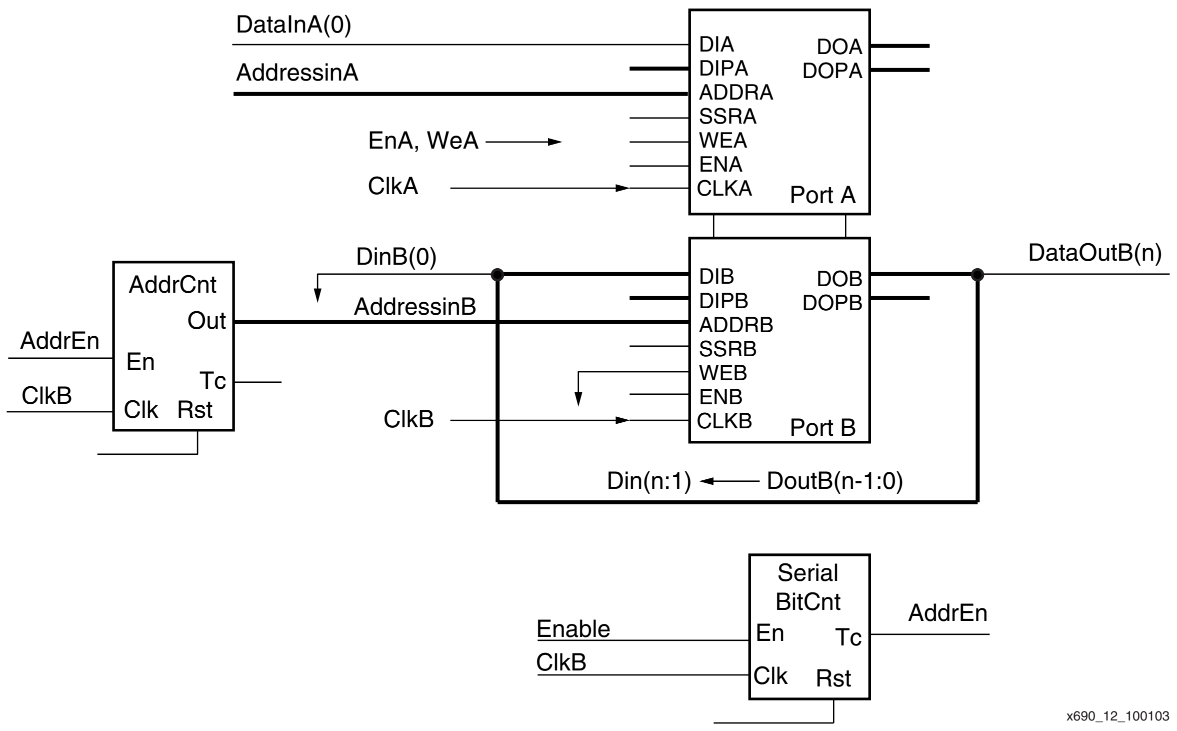

Figure 12: **Dual-Port Serializer**

A small counter running at the serial bitstream clock rate with the same clock as applied to port B of the RAM, determines when the address counter is incremented or decremented. A fully serial shifted data word is available at the terminal count of the bit counter, at the moment that the address changes state.

Dual-Port Serialization Implementation

The dual-port implementation of the parallel-to-serial converter in Block SelectRAM is limited to the fixed address implementation. Only one address place per Block SelectRAM port can be used, and there is no data storage capability. For the functionality and description, read the above described single-port implementation in "Single-Port Serialization Implementation," and duplicate it for both RAM ports.

This type of application typically is used for easy implementation of DDR applications (see Figure 13). Interleaved input data is presented to both multiplexers. When Load is one, data is stored in the RAM array. When Load is zero, the Block SelectRAM output is fed back through the multiplexer to the RAM input.

Odd bits are clocked in on the rising clock edge to port A, and even bits are clocked in on the falling clock edge to port B. Then the RAM output bits are used as serial outputs into a DDR I/O flip-flop, making it possible to double the serializer speed.

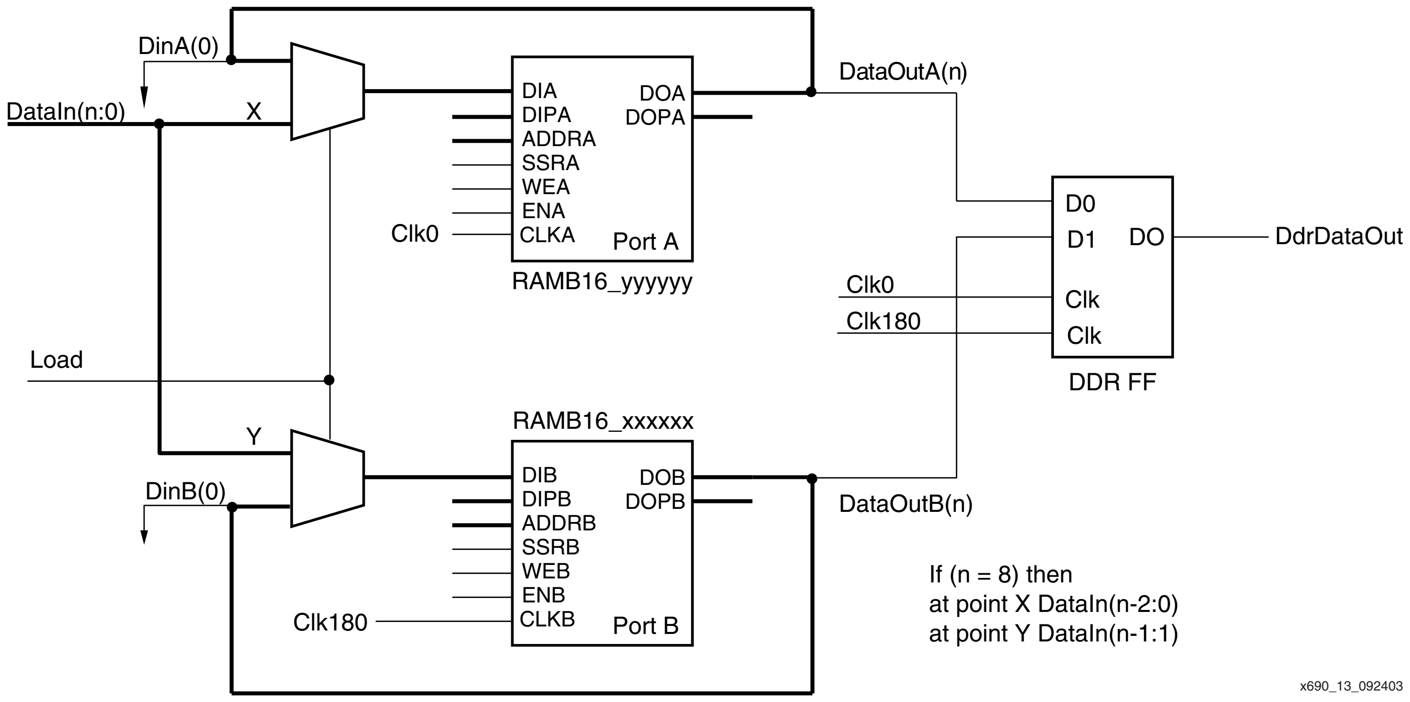

Figure 13: **Dual-Port Serializer with DDR Output**

Reference Design

Either design concept (serializing or deserializing) is suitable for use in either the Virtex-II, Virtex-II Pro or Spartan-3 families. The fully synthesizable design files in xapp690.zip are written in both VHDL and Verilog for both the single-port and dual-port cases described above.

Conclusion

Using the techniques described here, with some external logic for address decoding, the requirements for lower speed serial communication devices can be met efficiently using Block SelectRAM. Together with design ideas from existing Xilinx application notes (such as XAPP224, XAPP225), full-featured serial communication devices can be implemented efficiently in FPGA devices.

One example is where data/clock arrives differentially (LVDS). In this case, data is extracted from the stream and fed into the Block SelectRAM. When a certain, predefined, programmable pattern is recognized, data is passed or stored in the same Block SelectRAM for further treatment by another design level.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/06/03	1.0	Initial Xilinx release.