Parallelized convolutional interleaver implementation for efficient DDR memory access

Jennifer N. Downey, Mary Jo W. Shalkhauser, and Thomas P. Bizon
NASA Glenn Research Center 21000 Brookpark Road, Cleveland, Ohio 44135

ABSTRACT

Convolutional interleavers are used in many different communications systems to correct for burst errors due to atmospheric fades and scintillation. The interleaver size is related to the channel coherence time and the data rate. Small convolutional interleavers can be implemented in a field programmable gate array (FPGA) block random access memory (BRAM). However, large interleavers exceeding the size of the BRAM on the FPGA are necessary for channels with longer fades and higher data rates. Therefore, an implementation utilizing double data rate (DDR) memory external to the FPGA is necessary. Wide DDR memory data buses can make the use of DDR memory for convolutional interleavers inefficient when individual symbols are written to and read from the memory. DDR memory operational speeds can also limit the data rate of the interleaver. The Consultative Committee for Space Data Systems (CCSDS) Optical Communications High Photon Efficiency (HPE) standard utilizes a convolutional channel symbol interleaver. A previous implementation of the HPE standard utilized BRAM for the convolutional interleaver, but mission requirements for the upcoming Optical Artemis-2 Orion (O2O) communications demonstration dictate the use of an interleaver exceeding the size of the BRAM. An algorithm and method for implementing the convolutional interleaver in the FPGA with DDR memory is described in this paper.

Keywords: Optical communications, pulse position modulation, waveform, convolutional interleaver

1. INTRODUCTION

Convolutional interleavers are used in communications systems to compensate for burst errors. They are used in many different communications waveforms, including the Consultative Committee for Space Data Systems (CCSDS) Optical Communications High Photon Efficiency (HPE) Standard[1]. The CCSDS HPE standard is designed for photon-counting optical communication links, such as those from deep space. This standard is being used on the upcoming NASA missions to the Psyche[2] asteroid and the Optical Artemis-2 Orion[3] (O2O) communications demonstration. It is possible to implement some sizes of this interleaver in the field programmable gate array (FPGA) block random access memory (RAM) (BRAM) or by using shift registers. However, these NASA missions require a convolutional interleaver that exceeds the memory elements in the FPGA. The use of external DDR memory is necessary. If a symbol in the HPE transmit waveform is represented with 8 bits in the FPGA but the interface to DDR memory is much larger – 512 bits, it would be highly inefficient and too slow if only 8 bits out of the 512 bits are used in each read and write to the DDR. Therefore, a parallel method of interfacing to the DDR is necessary.

Background information on convolutional interleavers is described in Section 2. The parallel convolutional interleaver algorithm is described in Section 3. The FPGA implementation of the parallel convolutional interleaver is detailed in Section 4. This implementation can also be used in a convolutional deinterleaver with a small modification. The convolutional deinterleaver implementation (as well as an implementation using soft symbols) is described in Section 5. Utilization statistics are given in Section 6.

2. BACKGROUND

A convolutional interleaver can be realized with rows of varying-length shift registers, as shown in Figure 1. The size of a convolutional interleaver is described with parameters N and B. The value N determines the number of rows in the interleaver and the value B determines the length of each row. The first row is a pass through and has no delay. The second row has a length of B symbols. The third row has a length of $2B$ symbols, etc. The last row is of length $B(N - 1)$ symbols. Interleaving is performed by writing a symbol to the beginning of a row and reading a symbol from the end of the row. The next symbol is written to and read from the following row. Interleaving is performed by repeating this process in a circular fashion over all the rows in the interleaver.

When N=4 and B=8, the symbols into the deinterleaver $S_0, S_1, S_2, \ldots$ will produce an output of the following when the initial contents of the deinterleaver are set to zero:

$$T_0, T_1, T_2, T_3, T_4, T_5, \ldots T_{63}, \ldots = S_0, 0, 0, 0, S_4, 0, 0, 0, S_8, 0, 0, 0, S_{12}, 0, 0, 0, S_{16}, 0, 0, 0, S_{20}, 0, 0, 0, S_{24}, 0, 0, 0, S_{28}, 0, 0, 0, S_{32}, S_1, 0, 0, S_{36}, S_5, 0, 0, S_{40}, S_9, 0, 0, S_{44}, S_{13}, 0, 0, S_{48}, S_{17}, 0, 0, S_{52}, S_{21}, 0, 0, S_{56}, S_{25}, 0, 0, S_{60}, S_{29}, 0, 0, \ldots$$

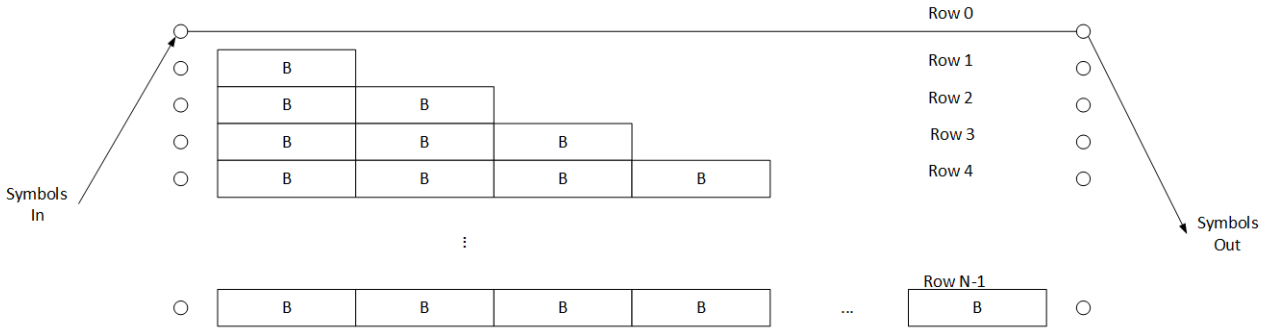


Figure 1. Diagram of a convolutional interleaver.

3. PARALLEL INTERLEAVER ALGORITHM DESCRIPTION

A single large interleaver of size (N, B) can be shown to be equivalent to P smaller interleavers of size (N, B/P) operating in parallel. The symbols into the interleaver are de-multiplexed into P streams, individually interleaved, and multiplexed back together into a single stream, as shown in Figure 2. In this algorithm, B must be an integer multiple of P.

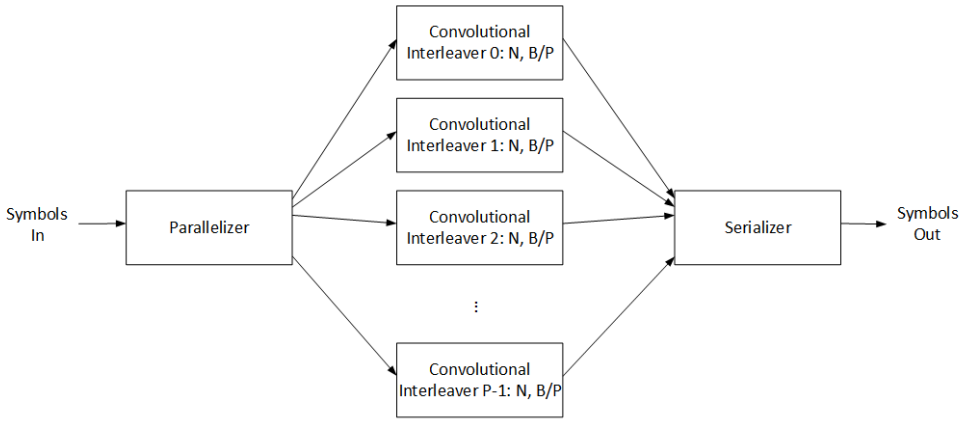


Figure 2. Diagram of a parallel convolutional interleaver with parameters (N, B/P).

3.1 Parallelizer

The parallelizer performs the de-multiplexing of the symbols into data blocks that are P rows by N columns. Each row in the data block is a queue, so that the first symbol written to the row is also the first symbol that will be read. Symbols are written in the order they are received until the first row is full, and then subsequent rows are filled. The data blocks are concatenated together, and each row is used as the input to a parallel convolutional interleaver with parameters (N, B/P).

The order of the symbols within the data blocks for a parallel interleaver with parameters N=4, B=8, and P=4 is illustrated in Figure 3. Each data block has P=4 rows and N=4 columns.

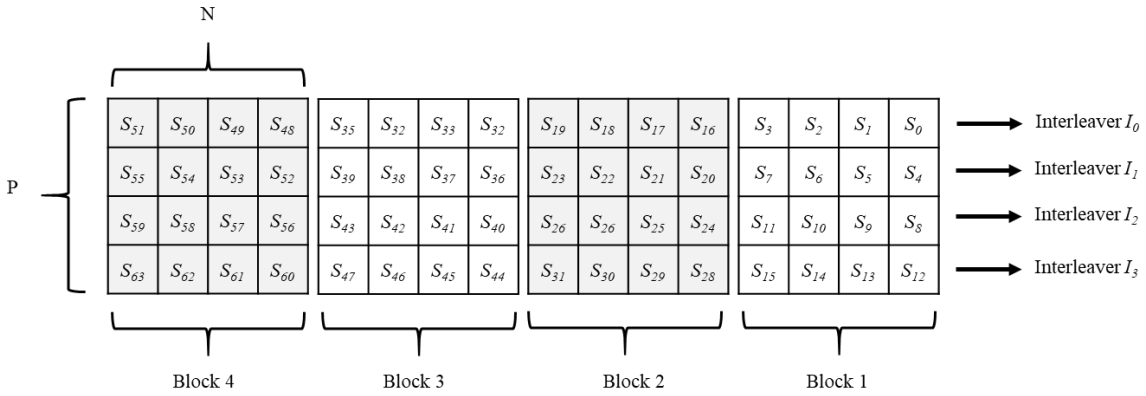


Figure 3. Diagram of the parallelizer. The input data symbols, S_i, are demultiplexed into to queues to form blocks of data that are P=4 rows and N=4 columns. The input to interleaver I_0 are symbols $S_0, S_1, S_2, S_3, S_{16}, S_{17}, S_{18}, S_{19}, S_{32}, \ldots$. The input to interleaver I_1 are symbols $S_4, S_5, S_6, S_7, S_{20}, S_{21}, S_{22}, S_{23}, S_{36}, \ldots$.

3.2 Serializer

The serializer multiplexes the symbols output from each parallel convolutional interleaver into a single stream. The symbols output from each parallel interleaver are written into data blocks of size P rows by N columns. Each row in the data block contains symbols from a different interleaver. Figure 4(a) shows how the output of each interleaver is organized into data blocks. The k^{th} symbol output from the j^{th} interleaver is denoted by $I_{j,k}$. When a data block is full, each row is read out of the data blocks and multiplexed into a single stream in the following order:

$$U_0, U_1, U_2, U_3, U_4, U_5, \ldots U_{63}, \ldots = I_{0,0}, I_{0,1}, I_{0,2}, I_{0,3}, I_{1,0}, I_{1,1}, I_{1,2}, I_{1,3}, I_{2,0}, I_{2,1}, I_{2,2}, I_{2,3}, I_{3,0}, I_{3,1}, I_{3,2}, I_{3,3}, I_{0,4}, I_{0,5}, I_{0,6}, I_{0,7}, I_{1,4}, I_{1,5}, I_{1,6}, I_{1,7}, I_{2,4}, I_{2,5}, I_{2,6}, I_{2,7}, I_{3,4}, I_{3,5}, I_{3,6}, I_{3,7}, I_{0,8}, I_{0,9}, I_{0,10}, I_{0,11}, \ldots I_{3,8}, I_{3,9}, I_{3,10}, I_{3,11}, \ldots$$

The output of the interleavers is shown in Figure 4(b) when the initial contents is set to zero. After serialization, the output symbols are in the same order as those of the single large interleaver:

$$U_0, U_1, U_2, U_3, U_4, U_5, \ldots U_{63}, \ldots = S_0, 0, 0, 0, S_4, 0, 0, 0, S_8, 0, 0, 0, S_{12}, 0, 0, 0, S_{16}, 0, 0, 0, S_{20}, 0, 0, 0, S_{24}, 0, 0, 0, S_{28}, 0, 0, 0, S_{32}, S_1, 0, 0, S_{36}, S_5, 0, 0, S_{40}, S_9, 0, 0, S_{44}, S_{13}, 0, 0, S_{48}, S_{17}, 0, 0, S_{52}, S_{21}, 0, 0, S_{56}, S_{25}, 0, 0, S_{60}, S_{29}, 0, 0, \ldots$$

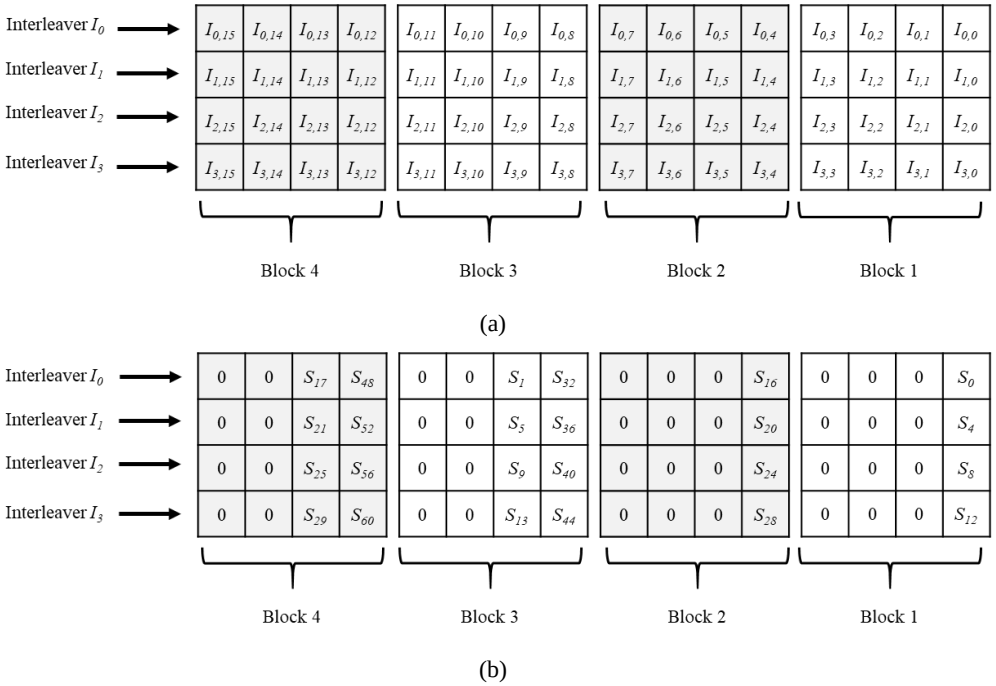


Figure 4. (a) Output of the parallel interleavers (N=4, B=2). Each row is the output of an interleaver. (b) When the interleavers are initialized to zero, the output from Interleaver I_0 is S_0, 0, 0, 0, S_{16}, 0, 0, 0, S_{32}, S_1, 0, 0, S_{48}, …, etc.

4. INTERLEAVER FPGA IMPLEMENTATION

The parallel interleaver was implemented in an FPGA using DDR memory. The parallelizer is implemented with a first-in, first-out buffer (FIFO) to assemble the *N*x*P* size blocks. Once a block is filled, each column in the block (shown in Figure 3) is written to an address in DDR, so that the same position in each of the *P* smaller interleavers is stored at the same DDR address. The DDR addresses are precomputed and stored in a ROM lookup table in the FPGA. In the serializer, data is read from the DDR, placed in a FIFO buffer, and multiplexed into a single output stream. A block diagram of this implementation is shown in Figure 5.

A parallel-18 (*P*=18) convolutional interleaver was implemented as a part of the CCSDS HPE transmit waveform in the FPGA with external DDR memory clocked at 1600 MHz. In this implementation, 8-bit symbols were used and the symbol rate for pulse position modulation (PPM)-16 was 100 MHz. Each DDR read/write utilized 144 out of the 512 available bits. Different size interleavers were configured using the same code by changing the lookup table addresses.

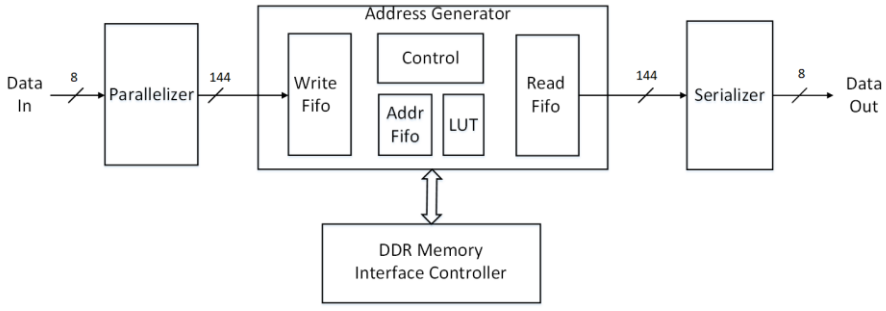


Figure 5. Hardware implementation of parallel convolutional interleaver, with parameters *N*, *B*, and *P*.

The parallelizer accepts 8-bit interleaved symbols and assembles them into data blocks. Symbols are written into the most significant bits (bits 143:136) of a 144-bit wide by *N* deep FIFO as they are received to create the first row of a *P*x*N* block as shown in Figure 3. When *N* symbols (i.e., the first row of the block) have been written into the FIFO, reads are started. The FIFO output is shifted by 8 bits and concatenated with the incoming 8-bit symbols, so that the incoming symbol is in the most significant 8-bits of the FIFO input. The next *N* symbols thus create the second row of the block. Once all 144 bits are filled with symbols (i.e., when 18*N* symbols have been written into the FIFO), data (the columns of each block) is output from the parallelizer to the DDR memory. A block diagram of the parallelizer is in Figure 6.

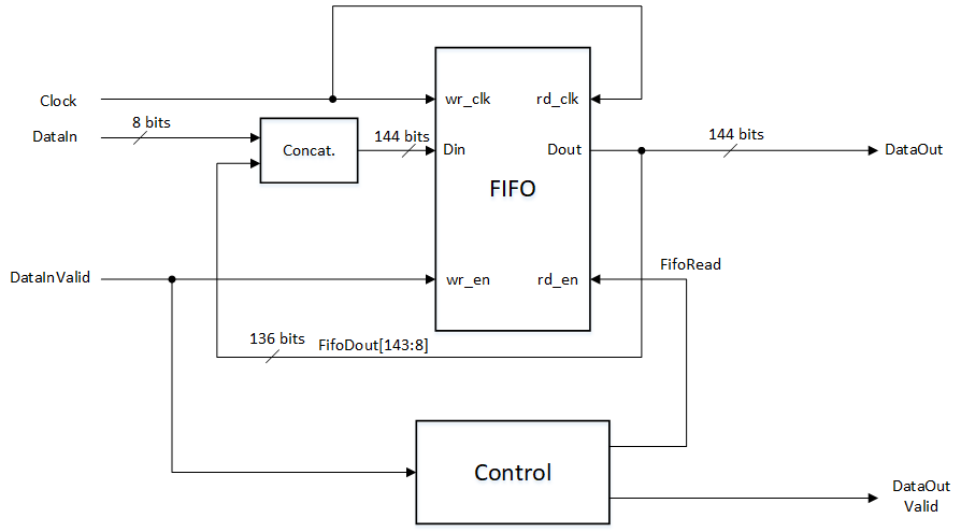


Figure 6. Parallelizer block diagram.

The address generator module follows the parallelizer and provides the data interface and the control signals for the DDR memory. A block diagram of the address generator is in Figure 7. In this module, data from the parallelizer is written into the write FIFO buffer. This buffer stores the data to be written into the DDR memory and crosses the clock domain from the interleaver clock domain to the DDR memory clock domain.

Interleaver rows are stored in consecutive DDR memory locations. The depth of these rows is determined by the value *B*/*P*. The starting address of each row is maintained in a ROM lookup table. The contents of the ROM lookup table are precomputed and contain starting addresses for multiple *N* and *B* pairs. The addresses for different *N* and *B* values can be accessed by addressing a different section of the ROM lookup table.

The lookup table is addressed by the row counter and outputs the starting address for each row of the interleaver. For the first group of *N* rows, input data is written to the starting address of each row – *P* parallel symbols per row. The address for the next group of *N* rows in the interleaver is incremented in the controller and stored in the address FIFO each time data is read from and written to the DDR. Once the address in the row becomes equal to the starting address for the next row, it is set back to the initial address for the row. When an address already contains a data value, the value is read from the memory before the new value is written.

Data is output from the DDR memory based upon the address applied to the memory. Output data is routed to the read FIFO buffer which stores the data and is used to cross the clock domain from the DDR memory domain to the interleaver clock domain.

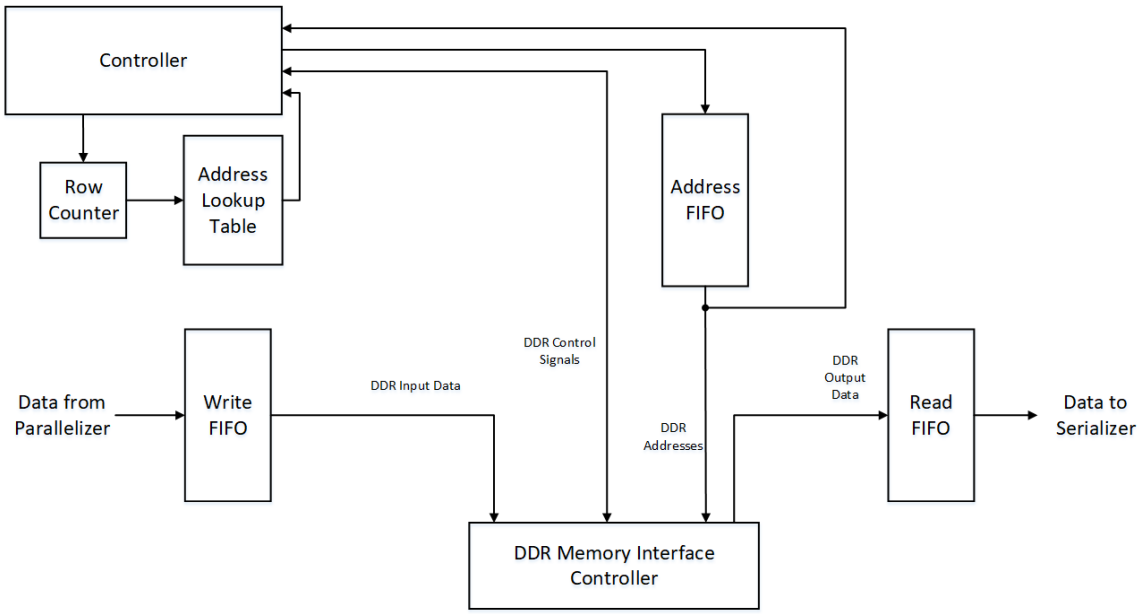


Figure 7. Address generator block diagram.

The serializer component accepts a 144-bit vector containing parallel groups of 18 symbols from the read FIFO in the address generator and writes them into the serializer 144-bit wide FIFO. When the FIFO contains a full data block (*N*x*P* symbols), FIFO reads start and the oldest symbol (bits 7:0) of each vector is output from the interleaver. The remaining FIFO read data (bits 143:8) is shifted right by one byte and fed back into the FIFO. As the last symbols are shifted out, the next set of 144-bit data is loaded into the FIFO and the process repeats. A block diagram of the serializer is in Figure 8.

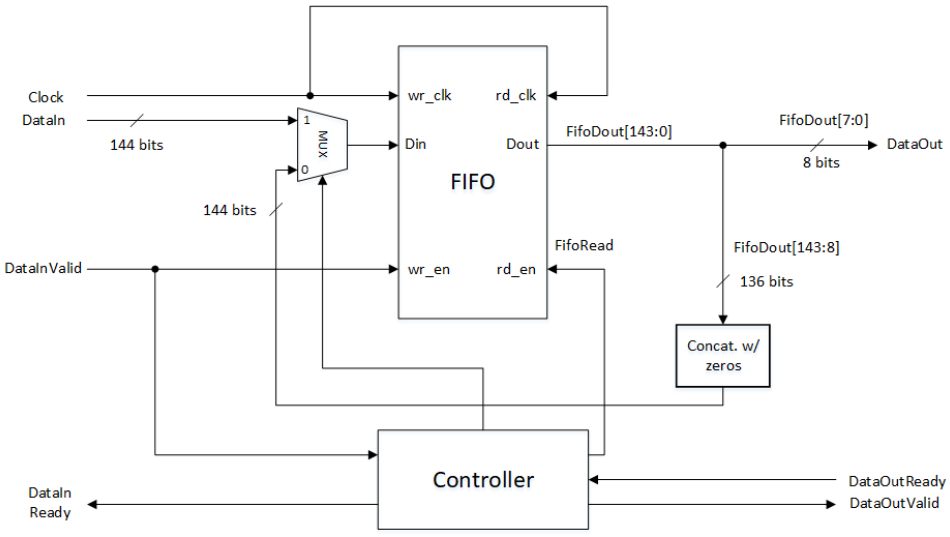


Figure 8. Serializer block diagram.

5. DEINTERLEAVER FPGA IMPLEMENTATION

A feature of this implementation is that an interleaver or a deinterleaver can be implemented by changing the DDR addresses in the lookup table. A parallel-18 convolutional deinterleaver was implemented using the same VHDL code as the interleaver by adding additional DDR addresses to the lookup table. The symbols in this implementation were derived via hard decision which resulted in a symbol size of 8 bits (the same symbol size as the transmitter). The external DDR memory was clocked at 1066 MHz, and the symbol rate for pulse position modulation (PPM)-16 was 100 MHz.

Many communication systems require soft decisions to be made on symbols, which results in additional bits per symbol. For example, with 3 bits per PPM slot, the symbol size is 48 bits for PPM-16 and 768 bits for PPM-256. These larger bit sizes required modifications to the original design of the interleaver and deinterleaver.

First, the change to soft symbols required an increase in the bus width to the DDR memory and a change in the number of parallel symbols. The original design with 18 parallel 8-bit symbols used 144 bits of the 512 bits available in the memory interface. Because of the larger symbol size, it was necessary to reduce the number of parallel symbols to 9 (48-bit symbols x 9 = 432 bits). The controllers for both the serializer and the parallelizer modules contain counters for the desired parallelism, and the new parallelization was implemented by changing the value loaded into the counter. The size of the parallelizer and serializer FIFOs were also increased to accommodate the need to store larger symbols prior to input into the DDR.

Second, multiple memory locations in the DDR were used to store a single symbol for modulation orders of PPM-32 and above. For example, two consecutive locations were used to store the parallel-9 group of PPM-32 symbols, and four consecutive locations were used to store the parallel-9 group of PPM-64 symbols. New lookup table addresses were generated for the parallel-9 implementation, and these addresses were scaled for higher PPM-orders with the addition of an inner loop counter and a scaling factor, as shown in Figure 9. This counter and scaling factor allows for multiple writes per parallel-9 group of symbols to DDR memory.

One final point is that with the number of parallel symbols reduced to 9, the DDR memory with a maximum rate of 1600 MHz on the current platform is not able to keep up with the highest desired slot clock of 2 GHz in the CCSDS HPE standard. This hardware will be replaced with faster DDR memory. Our analysis shows that DDR memory clocked at 2400 MHz will be needed for the CCSDS HPE standard 2 GHz slot clock modes. Optionally, two DDR memories clocked at 1600 MHz with a parallel-18 implementation could be used to achieve the needed performance.

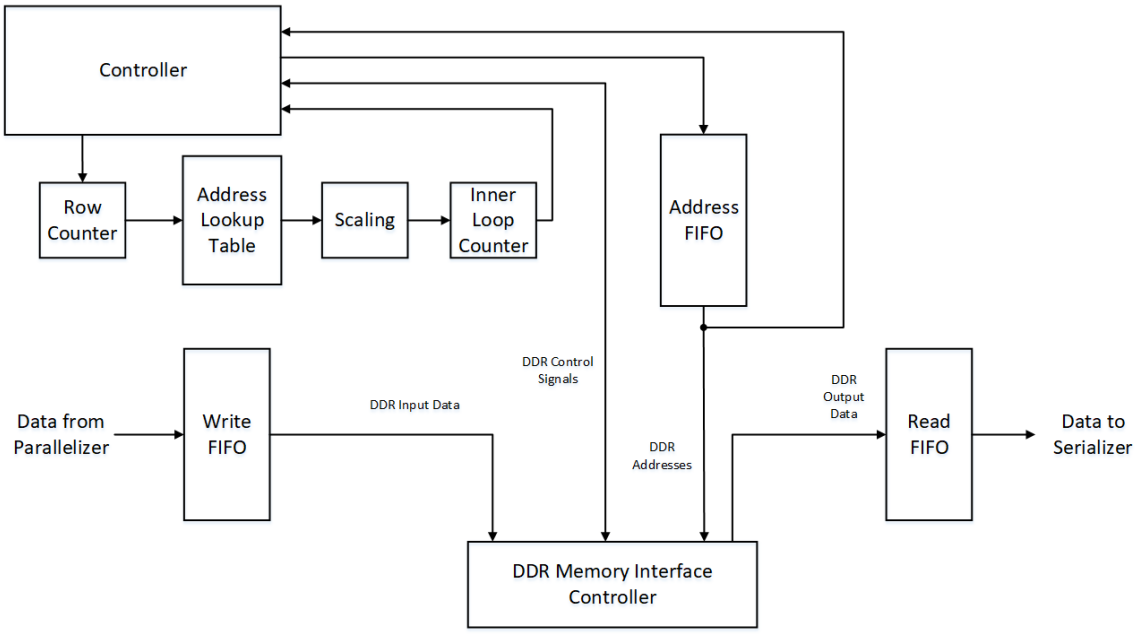


Figure 9. Soft symbol deinterleaver.

6. IMPLEMENTATION RESULTS

The interleaver was implemented with VHDL and integrated with a previous implementation[4] of the CCSDS HPE transmit waveform on a Xilinx VC707 FPGA development platform. The interleaver implements PPM orders 16, 32, 64, 128, and 256. It has been tested in hardware at the maximum symbol rates of 100 MHz for PPM-16 and 50 MHz PPM-32. The highest data rate tested with the interleaver is 267 Mbps with the 0.5 ns slot, PPM-16, code rate 2/3 mode. This implementation has been used to test the O2O space modem[5] receiver and will be used in the ground transmitter for the O2O White Sands ground station. FPGA utilization statistics for the interleaver are shown in Table 1. The utilization numbers show that this interleaver implementation overhead is low and can be included in any FPGA with an external DDR memory attached with minimal impact to the rest of the design.

Table 1. Interleaver FPGA utilization statistics.

Resource	Parallelizer	Address Generator	Serializer	DDR Memory Interface Controller	Total	% Utilization
Slice LUTs	83	293	234	10,406	11,016	3.6
Slice Registers	304	753	304	6,782	8,143	1.3
Muxes	0	0	0	219	219	0.1
Slices	73	214	97	3,451	3,835	5.1
LUT as Logic	83	293	237	8,936	9,549	3.1
LUT as Memory	0	0	0	1,470	1,470	1.1
BRAM	2	8	2	0	12	1.2

The hard and soft symbol deinterleavers have been both implemented in VHDL and tested in an FPGA. The hard symbol deinterleaver has been used in a receiver designed to check for compatibility of the received signal with the CCSDS HPE standard. The deinterleaver has been tested in the FPGA at the maximum symbol rates of 100 MHz for PPM-16 and 50 MHz for PPM-32. The hard symbol deinterleaver utilizes the exact same code as the interleaver, but with different memory addresses programmed into the address lookup table. The utilization statistics are the same as the interleaver. The soft symbol deinterleaver has also been implemented in VHDL and integrated with a full CCSDS HPE photon counting receiver currently in development[6]. The soft decision deinterleaver has been tested at a symbol rate of 50 MHz for PPM-16 and 25 MHz for PPM-32. The highest data rate tested with the soft symbol deinterleaver is 83 Mbps with the 1 ns slot, PPM-32, code rate 2/3 mode. Utilization statistics for the soft symbol deinterleaver are given in Table 2. Both receivers are currently implemented on a Vadatech AMC516 FPGA card with a FMC211 mezzanine card for the analog to digital converter. The soft decision deinterleaver rate is limited by the DDR memory speed, and higher rates will be demonstrated with new hardware in the future.

Table 2. Soft symbol deinterleaver FPGA utilization statistics.

Resource	Parallelizer	Address Generator	Serializer	DDR Memory Interface Controller	Total	% Utilization
Slice LUTs	126	481	587	16,025	17,219	4.0
Slice Registers	693	1,537	687	11,705	14,622	1.7
Muxes	0	0	0	188	188	0.1
Slices	221	570	367	5,365	6,523	6.0
LUT as Logic	124	477	585	13,892	15,078	3.5
LUT as Memory	2	4	2	2,133	2,141	1.2
BRAM	27	59	27	0	113	7.7

7. CONCLUSION

An algorithm for using *P* parallel convolutional interleavers multiplexed together was designed and has been shown to be equivalent to a large convolutional interleaver. This algorithm was implemented in an FPGA and uses external DDR memory. The design takes up very little space in the FPGA, leaving significant room for other parts of the design. The parallel interleaver parameters *N* and *B* can be modified just by changing the lookup table in the FPGA. No code modifications are required. This interleaving method can also be directly applied to a convolutional deinterleaver. The same code can be used for both an interleaver and a hard decision deinterleaver, with only changes to the addresses stored in the lookup table. A soft decision deinterleaver was also implemented in the FPGA with minor changes to the design.

REFERENCES

[1] "Optical communications coding and synchronization recommended standard," CCSDS 142.0-B-1 Blue Book (2019).

[2] Biswas, A., Srinivasan, M., Piazzolla, S., and Hoppe, D. "Deep space optical communications," Proc. SPIE Free-Space Laser Communication and Atmospheric Propagation XXX 10524 (2018).

[3] Robinson, B., Shih, T., Khatri, F., Boroson, D., Burnside, J., Guldner, O., Constantine, S., Torres, J., Yarnall, T., DeVoe, C., Hubbard, W., Geisler, D., Stevens, M., Mikulina, O., Spellmeyer, N., Wang, J., Butler, R., Hogan, M., King, T., and Seas, A. "Laser communications for human space exploration in cislunar space: ILLUMA-T and O2O," Proc. SPIE Free-Space Laser Communication and Atmospheric Propagation XXX 10524 (2018).

[4] Nappier, J. and Lantz, N. "Development of an optical slice for an RF and optical software defined radio," Proc. SPIE Free-Space Laser Communication and Atmospheric Propagation XXX 10524 (2018).

[5] Dailey, J., Koch, T., Delatore, J., Kolchmeyer, J., Husaini, S., Dinu, M., Le Grange, J. Yagudayev, B., Stenard, A., Crandall, L., Szczepanik, G., Wendt, N., Monte, A., Wood, T., Schulein, B., and Geisler, D. "Optical modem enabling broadband datacom links for crewed cos-lunar missions," Proc. SPIE Free-Space Laser Communication and Atmospheric Propagation XXXIV 11993 (2022).

[6] B. E. Vyhnalek, J. M. Nappier and S. A. Tedder, "Real Time Photon-Counting Receiver for High Photon Efficiency Optical Communications," IEEE International Conference on Space Optical Systems and Applications (ICSOS), pp. 1-6, (2019).