A HARDWARE ARCHITECTURE FOR A MULTI MODE BLOCK INTERLEAVER

Eric Tell, Dake Liu

Dept. of EE, Linköping University, SE-581 83 Linköping, Sweden
{erite, dake}@isy.liu.se

***Abstract* - We are interested in developing a programmable baseband processor for software defined radio and are trying to find configurable hardware blocks that can be used in multiple radio standards, including for example wireless LAN and 3G standards.**

This paper suggests an architecture for a multi mode block interleaver that is suitable e.g. for the IEEE 802.11a and 802.11g standards. Our implementation is based on a special matrix memory to which data is written as rows but read out as columns.

To enable a comparison, an interleaver for the Wireless LAN standard 802.11a has been implemented both using our suggested architecture and using a traditional interleaver implementation based on a bit memory.

Our implementation reaches a significantly higher performance and a lower power consumption with no extra area. The price to pay is a small loss of generality.

I. Introduction

We want to develop a platform for software defined radio based on a programmable processor supported by configurable hardware accelerator blocks for functions that are not well suited for software implementation. Typical examples of such functions are algorithms operating on bit level, such as channel coding functions (FEC, interleaving).

This paper introduces a new architecture for a configurable multi mode block interleaver. As opposed to traditional block interleaver implementations where only one bit is read or written per clock cycle, in the new architecture larger words are read/written in parallel. This leads to a significant performance increase as well as decreased power consumption. The new architecture can also be implemented to match the memory interface of a host processor.

Section II of the paper gives some background on block interleaving. Section III gives an example of an interleaving scheme that is used in several OFDM based wireless LAN standards. Section IV describes a traditional block interleaver implementation and section V introduces the proposed new architecture. Differences between the two schemes are discussed in section VI and implementation results are found in section VII.

II. Background

Interleaving is used in radio systems to distribute transmitted bits in time or frequency or both. Consecutive bits on the input stream should not be transmitted consecutively (or on the same frequency in an OFDM system). Interleaving reduces the effect of burst errors caused by a fast fading (or frequency selective fading in an OFDM system) channel [1][2].

One major class of interleavers are block interleavers. A block interleaver operates on one block of input bits at a time and there is no interleaving between the blocks. A block interleaver is often implemented by writing the bits into a matrix row by row and then reading them column by column. Deinterleaving is simply the reversed operation - write column by column and read row by row.

In a general block interleaver the number of rows and columns must be configurable. However, a survey of interleaving schemes for different radio standards shows that this is not enough. A more general interleaver should be able to carry out the following steps:

1. Write bits to matrix row by row.
2. Perform intra-row permutations.
3. Perform intra-column permutations.
4. Read bits column by column.

In addition, some standards (including IEEE 802.11a) use different permutations for different rows/columns).

III. Interleaving for OFDM Wireless LANs

The wireless LAN standards 802.11a [3], 802.11g and Hiperlan/2 [4] are all OFDM based and use 48 data subcarriers and 4 pilot tones. They also use the same interleaving scheme.

The interleaving scheme is defined by two permutations. The first permutation ensures that adjacent bits are modulated onto nonadjacent subcarriers and the second permutation ensures that that adjacent bits are mapped alternatively onto less and more significant bits of the constellation, thereby long runs of low reliability (LSB) are avoided. The block size is always equal to one OFDM symbol and depends on the signal constellation.

In the equations below, k denotes the index of the bit before the first permutation, i denotes the index after the first but before the second permutation and j denotes the index after the second permutation. N is the number of bits in an OFDM symbol and N_{BPSC} is the number of bits per subcarrier. The two permutations are given by:

$$i = (N_S/16)(k \bmod 16) + floor(k/16) \tag{1}$$
$$j = s * floor(i/s) + (i + N - floor(16i/N)) \bmod s \tag{2}$$
$$s = max(N_{BPSC}/2, 1)$$

Closer study reveals that the first permutation defines a block interleaver with 16 columns and a variable number of rows. The second permutation defines intra-column permutations which depend on the block size and may also differ between columns. Tables 1 and 2 show the different block sizes and permutation schemes used in this scheme.

Constellation	**Bits/subc.**	**Block size**	**Rows**
BPSK	1	48	3
QPSK	2	96	6
16-QAM	4	192	12
64-QAM	6	288	18

Table 1: Block size for different signal constellations

Constellation	**Intra-column permutation**
BPSK	no permutation
QPSK	no permutation
16-QAM	$< 1, 0, 3, 2, 5, 4, 7, 6, 9, 8, 11, 10 >$
64-QAM	Three different permutations are used, alternating every column

Table 2: Intra column permutations for different signal constellations

This is one example of how many block interleaving schemes can be expressed in terms of the four steps described in section II.

IV. Traditional Interleaver Implementation

Figure 1 describes a traditional block interleaver implementation based on a bit addressable memory and a ROM (or look-up table) storing the interleaving sequence.

This implementation is completely general and even supports interleaving schemes that cannot be described by the steps outlined in section II. For interleaving operation the input bits are first written sequentially to the memory and then read in the order defined by the LUT. For deinterleaving the bits are written according to the LUT and then read sequentially.

The number of bits in the data memory is equal to the largest block size and the number of words in the LUT is at least equal to the sum of the block sizes of all interleaving schemes it should handle. For example in the 802.11a interleaver the data memory size is 288 (maximum block size) bits and the LUT needs 624 (sum of all block sizes) 9-bit entries.

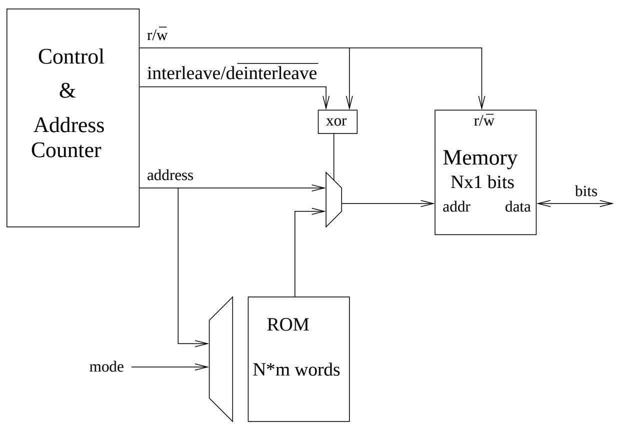

Figure 1: Traditional block interleaver implementation

The main advantages of this implementation are that it is very simple and straight forward and that it is completely general as long as the block size is fixed for each mode. (Schemes where the number of rows is arbitrary are not well supported.

The main disadvantage is that the bits have to be written and read one at a time.

Another disadvantage is that the complete interleaving sequence of every interleaving scheme has to be stored explicitly in the LUT, making it relatively large if many modes have to be supported.

V. A Novel Interleaver Architecture

The motivation for the new implementation is to try to avoid the need for addressing each individual bit in the memory and instead enable read and write of several bits in parallel. The new architecture should also be more suitable for operation as an accelerator unit for a programmable processor. It should therefore be possible to make the interleaver compatible with the processor word length. Ideally the interleaver interface to the processor should look like an ordinary memory. Then the interleaver could be used as a normal data buffer and it would be possible to carry out the interleaving at no extra instruction cost.

The number of rows and columns should be configurable. The solution should be suitable for multiple standards, i.e. multiple intra-row and intra-column permutation schemes should be supported. It might not have to be completely general but preferably the cost for supporting additional modes should be small compared to the corresponding cost (for increasing ROM size) in the traditional approach.

Figure 2 shows the main idea of the new architecture. The architecture is based on a special matrix memory block where words are written as rows but read as columns.

A complete row can be written in one clock cycle and a complete column can be read. Intra-row permutations are carried out before the bits are stored to memory by simply reordering the bits on the input data bus. In the

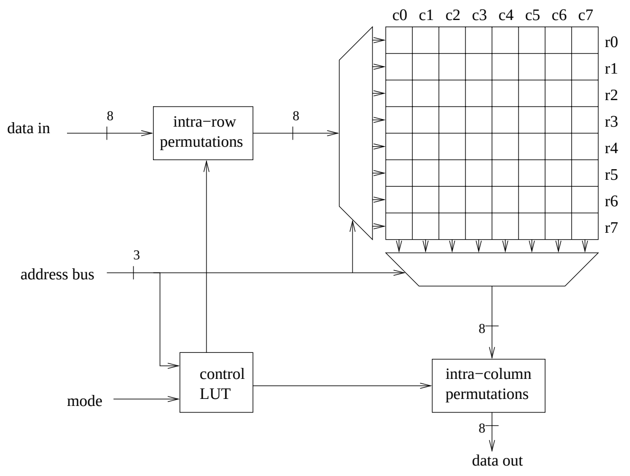

Figure 2: New interleaver implementation

same way intra-column permutations are carried out by reordering the bits on the output data bus after the data has been read. If a small number of different permutation schemes are needed, the permutation blocks are just a set of small multiplexers using the same control signal.

The operation for deinterleaving is the same as for interleaving. The deinterleaving is seen as just another interleaving scheme where the row number has become column number and the intra-row permutations have become intra-column permutations and vice versa.

The matrix is dimensioned according to the largest number of rows or columns that has to be supported. If an interleaving scheme with smaller block size is used the extra rows and/or columns are deactivated to save power.

The interface of the interleaver can be made compatible with the memory interface of a processor. Hence it may be possible to use the interleaver as a computing buffer, as described above.

Since the addressing is taken care of by the processor, very little control logic is needed in the interleaver itself.

The interleaver is configured by three control signals specifying the number of rows, the number of columns and the permutation scheme to use.

A control block may be needed to decide the permutation mode according to the current mode (and address). This is implemented as a small look-up table.

VI. Comparison of the two schemes

The main constraint for the new architecture is the decreased flexibility compared to the completely general traditional block interleaver implementation. This is however compensated by large gains in performance and power consumption. Furthermore the flexibility is still enough for the standards we have studied.

VI.1. Performance

Using the new architecture, it is possible to interleave a block of R rows and C columns in $R + C$ clock cycles, as long as both R and C are smaller than the longest word length of the processor. Generally the cycle count is $R * ceil(C/l) + C * ceil(R/l)$, where l is the processor word length.

However additional processing will be needed in most cases since R and C are generally not equal to processor word length. The extra cycle count depends on the support for bit field operations in the processor and on the implementation of other baseband algorithms.

Nevertheless the performance is significantly increased compared to the traditional implementation which needs $2 * R * C$ cycles.

VI.2. Hardware Cost

The number of memory cells needed in the traditional implementation is equal to the largest block size that should be supported. For the new architecture the number of rows/columns in the matrix must both be equal to the largest number of rows/columns that should be supported (Note that these two values may not come from the same mode). When the same hardware is used for both interleaving and deinterleaving, both the number of rows and the number of columns will be equal to the largest number of rows *or* columns supported (because the role of rows and columns are swapped for deinterleaving). This means that the number of memory cells (bits) is larger in the new architecture, but still in the same order of magnitude. This increase is however compensated by the fact that the address decoding logic is much smaller in the new scheme due to the smaller number of addresses.

A considerable part of the area in the old architecture is occupied by the LUT, which needs one entry per bit. The corresponding hardware in the new architecture only needs *at most* one entry per row and column to store permutation modes (often the same mode is used for *all* rows/columns). This means that the extra area for each scheme that needs to be implemented is much smaller in the new architecture as long as the permutation blocks do not become significantly larger.

The weak link is the implementation of the permutation blocks. Implementing a small number (such as 8 or 16) of fixed patterns is no problem. E.g. in the implementation described in section VII each permutation block consists of 18 4-to-1 multiplexers. However, attempts to make a more general permutation block (something like a general crossbar switch) to support future standards will cause problems both in terms of critical path, area and control.

VI.3. Power consumption

The new architecture is not only superior when it comes to speed. It will also consume significantly less power. There are several reasons for this:

1. Most of the energy is consumed in the data memory. The new architecture needs more memory cells than the traditional one, but since rows/columns that

are not needed are deactivated, the number of *active memory cells* are the same in both solutions. Furthermore the address decoding is much simpler in the new architecture since only one address per row/column is needed instead of one per bit. Hence the power needed for decoding each bit is smaller.

2. The ROM holding the interleaving sequence in the old scheme consumes significant power. It becomes rather large if multiple interleaving schemes have to be supported. (The control LUT in the new architecture is very small compared to the sequence ROM)

3. More power is also consumed, for example in the control logic, in the old scheme simply because it has to run for many more clock cycles.

4. Since the new architecture completes the interleaving in fewer clock cycles it can run at a lower clock frequency. This may make it possible to reduce power consumption by lowering the supply voltage.

Again the place where the new implementation may lose power is if the permutation units get very complex.

VII. Implementation Results

An interleaver/deinterleaver for 802.11a was implemented using both the new and the traditional architecture. This interleaver handles four different schemes, as described in section III The block size is up to 16x18=288 and four different row/column permutation schemes are needed.

The two architectures were implemented in VHDL and synthesized for a 130 nm standard cell library from UMC. The data memory cells are based on latches from the standard cell library. A more efficient solution may be possible using e.g. an SRAM-like approach (not achievable using standard cell design). However, our implementations still demonstrate the difference between the two architectures. Furthermore the advantages of using special memory design techniques are limited for memories as small as these. The implementation results are shown in table 3.

Feature	**old arch.**	**new arch.**
BPSK cycle count	96 cycles	19 cycles
64-QAM cycle count	576 cycles	34 cycles
Approx. max freq.	250 MHz	500 MHz
Data memory area	0.0136 mm^2	0.0120 mm^2
Other area	0.0076 mm^2	0.0021 mm^2
Total area	0.0212 mm^2	0.0141 mm^2

Table 3: 802.11a interleaver implementation results

No pipelining was used at all. This gives a very long critical path for the old architecture (from the address counter through both sequence memory and data memory). With appropriate pipelining the two implementations should run at approximately the same clock frequency.

Note that the data memory area is smaller for the new architecture although it has more cells (18x18 vs. 16x18). This is due the large address decoder in the old architecture.

The difference in area not occupied by data memory is due to the sequence memory (ROM; 0.0064 mm^2) in the old architecture.

VIII. Summary and Conclusions

Two ways of implementing interleavers have been presented. In the traditional bit memory based interleaver, bits are written to a sequential memory one at a time and then read out one at a time in a different order. A look-up table stores the read-out order.

The new implementation presented is based on a special matrix memory into which complete rows/columns can be read/written in one clock cycle.

The new implementation has a much higher performance, lower power consumption and is also more suitable for use together with a programmable processor. However it is not as general as the traditional implementation for which any interleaving scheme can be implemented by changing the contents of the LUT.

The advantage of using the new implementation is largest if only a small number of previously known row and column permutations are needed. In that case the area is about the same as for the traditional implementation although the performance and power consumption is a lot better.

References

[1] H Heiskala & J T Terry, *OFDM Wireless LANs: A Theoretical and practical guide*, Sams Publishing, 2002.

[2] Shu Lin & Daniel J Costello Jr, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, 1983

[3] IEEE 802.11a, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications High-speed Physical Layer in the 5 GHz Band*, 1999.

[4] ETSI TS 101 475, *Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer*, 2001.