Efficient Implementation of FPGA to Generate the Address for Various Code Rates/Modulation Scheme of Wimax Deinterleaver without Manual Computation

Sagar. C 1*
1 P.G.Scholar, ECE Dept., PES College of Engg., Mandya, VTU, India

Nanda. B. S 2
2 Asst Prof., ECE Dept., PES College of Engg., Mandya. VTU, India

Abstract—

The main aim of this paper is to reduce floor function concept which is very difficult to implement Field Programmable Gate Array (FPGA) associated with the standards of IEEE 802.16. So instead we use a simple mathematical steps of algorithm in order to generate an address for WiMAX Deinterleaver so that low power and novel technique is obtained using Xilinx Field Programmable Gate Array (FPGA). Here in order to generate relevant address, we make use of three different modulation schemes of Quadrature like QPSK, 16-QAM 64-QAM 64 along with their code rates.

Keywords— Deinterleaver/Interleaver circuits, Modulation Circuits, Wireless Systems.

I. INTRODUCTION

As we know, IEEE has already developed standards for mobile broadband wireless access which is by term popularly referred as mobile WiMAX [1]. So the channel interleaver used with trans receiver of WiMAX plays a major role in minimizing effect of burst errors [2] which causes error in output LUT.

In this brief, low complexity, high speed, novel approach with resource efficient technique can generate address for channel deinterleaver by eliminating the floor function complexity. We can see a very few works are available in literature related to implementation of hardware. The work in [3] depicts the grouping of incoming data into the block in order to reduce frequency of memory access in Deinterleaver using conventional Look Up Table (LUT) concept for generating address for WiMAX. A low cost and reconfigurable architecture would be the added advantage for computation of required address. In communication industry, an IEEE 802.16e [4] called WiMAX is used with many variations like different on modulation schemes or different in code rates. System level overview of WiMAX along with different blocks is shown in fig 1.

Fig. 1. Block diagram of the WiMAX transreceiver.

In the interleaver block, data is written sequential order in to a memory whereas after applying certain permutations, the data is read in random manner. The block interleaver can also be considered as row-column matrix. Some approaches of work is been carried out in [2]-[4] on different scenarios but unfortunately, no work has been found related to hardware implementation. So this paper emphasizes in complexity reductions of generating address in 2D transformation of main interleaving functions. Modelsim is used for software simulation process in order to verify functionality of proposed algorithm and hardware. Also comparison is been made with recent work and the past work. An improvement in performance is obtained using embedded multiplier block, with minimization of interconnection delay. Hence overall a low complexity with efficient utilization of resource has made it wise to obtain 49% of betterment in terms of maximum operating frequency when compared with LUT technique. Also when compared with complicated and lengthy expressions especially with QAM techniques, the LUT based approach confirms the superiority of our proposed design.
in order to make the design compact, we can optimize the design by sharing common modules used in three different modulation schemes and finally embed all those schemes used into a single compact architecture. The architecture we use here can be modelled using Very High Description Language (VHDL) and is implemented using Spartan-3 kit version of FPGA. This document is a template. An electronic copy can be downloaded from the Journal website. For questions on paper guidelines, please contact the conference publications committee as indicated on the conference website. Information about final paper submission is available from the conference website.

II. WiMAX TRANSRECEIVER SYSTEM

Now let us know about how the working principle carried out in WiMAX system. As we can see the mandatory blocks used in fig 1, the incoming data streams obtained from the source part is been encoded using two main Forward Error Correction (FEC) coding techniques and before this, randomization of data streams procedure will be carried out. By this process, effect of burst error is minimized using the permutation technique to the received data stream. Actually there is no need of channel interleaver part when Convolutional Turbo Code (CTC) is used. Because the interleaver part is inbuilt in CTC itself [7]. Now when we look into the basic block diagram as shown in fig 1, there the mapper and inverse Fast Fourier Transform (FFT) block is used in order for modulation and construction of Orthogonal Frequency–Division Multiplexing (OFDM) process. Now in order to obtain the original data sequence at the output part, reverse order of restoration is been done.

When we look into fig 2 as shown, we can observe the read and write operation done in the address generator block. Since any one operation that is either read or write will be enable once at a time, we make use of multiplexer with select lines as shown. Observe that not gate is used in order to switch from one to the other operation. Here we can see that two memory blocks are used one for read and one for write operation where whenever the select line=1, Write enabled signal M-1 will be active and vice versa. Finally the interleaved data will be obtained as the output as seen by fig 2.

In block interleaver shown in fig 3, input symbols are written sequentially row by row and the output symbols are obtained by reading the column sequentially until the interleaver is emptied. While the interleaver is emptied, it is loaded again and the cycle repeats. At the receiver before decoding, the received bits are deinterleaved to get the original encoded data. This is in the form of M*N array where N is length of the code word.
A. SIMPLE MATHEMATICAL EXPRESSIONS FOR INTERLEAVER PART.
As we have stated that instead of floor function we can replace it by the simple mathematical steps, here in interleaver part, we use the following expressions. The two sets of expressions depicted in (1) and (2) ensure mapping of coded bits onto non adjacent subcarriers and alternate the less or most significant bits of the constellation used in modulation process respectively.

\[ m_k = \left( \frac{N_{cbps}}{d} \right) \cdot \left( k \% d \right) + \left\lfloor \frac{k}{d} \right\rfloor \]  \hspace{1cm} (1)

\[ j_k = s \cdot \left\lfloor \frac{m_k}{s} \right\rfloor + \left( m_k + N_{cbps} - \left\lfloor \frac{d \cdot m_k}{N_{cbps}} \right\rfloor \right) \% s. \]  \hspace{1cm} (2)

Here, \( d \) represent the number of columns used, \( m_k \) and \( j_k \) represent the outputs with respect to the first and second steps. \( K \) varies from 0 to \( N_{cbps} - 1 \). \( s \) is a parameter which is define as \( s = N_{pc} / 2 \) wherein \( N_{pc} \) is number of coded bits per subcarrier used in three different modulation schemes. \( \% \) and \( | \) sign are used for floor function and modulo respectively. By using these two important expressions, we can obtain the depth and code rates for all three modulation schemes. Similarly we can obtain the depths for various code rates for Deinterleaver part too and that mathematical expression has been shown below.

B. SIMPLE EXPRESSIONS FOR DEINTERLEAVER PART.
Similarly as in interleaver part, we make use of simple mathematical steps for deinterleaver part too. As shown in (3) and (4), let \( m_j \) and \( k_j \) defines the first and second level of permutation for the deinterleaver where \( j \) is the index of bits received. Here these two expressions of deinterleaver part perform the exact inversion of the expressions used in interleaver part.

\[ m_j = s \cdot \left\lfloor \frac{j}{s} \right\rfloor + \left( j + \left\lfloor \frac{d \cdot j}{N_{cbps}} \right\rfloor \right) \% s \]  \hspace{1cm} (3)

\[ k_j = d \cdot m_j - (N_{cbps} - 1) \cdot \left\lfloor \frac{d \cdot m_j}{N_{cbps}} \right\rfloor. \]  \hspace{1cm} (4)

III. ALGORITHMS USED TO GENERATE ADDRESS
Here, the proposed algorithm for generating the address for WiMAX Deinterleaver along with its mathematical background is been described. A MATLAB program is developed using (3) and (4) for all permissible code rates and modulation schemes. Now in order to generate the address for three modulation schemes, certain procedure is followed according to the algorithm. Let us look into the table below, where the required address can be generated for the first four rows (each four rows are meant for each modulation type used) and five columns as shown below.

<table>
<thead>
<tr>
<th>( N_{cbps} ) code rate and modulation type</th>
<th>De-interleaver addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N_{cbps} = 96 \text{-bits, } \frac{1}{2} \text{ code rate, QPSK} )</td>
<td>0 16 32 48 64</td>
</tr>
<tr>
<td></td>
<td>1 17 33 49 65</td>
</tr>
<tr>
<td></td>
<td>2 18 34 50 66</td>
</tr>
<tr>
<td></td>
<td>3 19 35 51 67</td>
</tr>
<tr>
<td>( N_{cbps} = 192 \text{-bits, } \frac{1}{2} \text{ code rate, 16-QAM} )</td>
<td>0 16 32 48 64</td>
</tr>
<tr>
<td></td>
<td>17 1 49 33 81</td>
</tr>
<tr>
<td></td>
<td>2 18 34 50 66</td>
</tr>
<tr>
<td></td>
<td>3 19 35 51 63</td>
</tr>
<tr>
<td>( N_{cbps} = 576 \text{-bits, } \frac{1}{4} \text{ code rate, 64-QAM} )</td>
<td>0 16 32 48 64</td>
</tr>
<tr>
<td></td>
<td>17 33 1 65 81</td>
</tr>
<tr>
<td></td>
<td>34 2 18 82 50</td>
</tr>
<tr>
<td></td>
<td>3 19 35 51 67</td>
</tr>
</tbody>
</table>

So address is generated for all the modulation type with different code rates and bit size. Here we choose \( d = 16 \) and the number of rows are fixed for all \( N_{cbps} \), whereas to obtain the number of columns, we divide \( N_{cbps} / d \) that is we choose as \( 96 / 16 = 6 \) bits. Now the table below shows how the mathematical foundation of the correlation between the address is achieved. It is achieved using certain expressions as derived in this brief if represented by (5)-(7). Where \( j=0,1,..,d-1 \) and \( i=0,1,.., (N_{cbps}/d)-1 \) represent row and column address.
The above algorithm shows how the address is obtained for the QPSK modulation scheme. After the initialization step, for loop is used as same as how we use in C programming and later the value of j and i is incremented as shown and then it is multiplied with d (the number of columns used). And that result is added with j value. The similar steps of algorithm are used for the other two modulation type and hence as a result, their address will be obtained. The above algorithm shows how the address is obtained for the QPSK modulation scheme. After the initialization step, for loop is used as same as how we use in C programming and later the value of j and i is incremented as shown and then it is multiplied with d (the number of columns used). And that result is added with j value. The similar steps of algorithm are used for the other two modulation type and hence as a result, their address will be obtained.

Algorithm for QPSK,

\[ k_{n,\text{QPSK}} = \{ d \cdot i + j \text{ for } \forall j \text{ and } \forall i \]  

Initialize Ncbps and d

for j=0 to d-1, j++

for i=0 to (Ncbps/d)-1, i++

\[ k_n = d \cdot i + j \]

end for

end for

Algorithm for 16-QAM:

\[ k_{n,\text{16-QAM}} = \begin{cases} 
  d \cdot i + j & \text{for } j \% 2 = 0 \text{ and for } \forall i \\
  d \cdot (i+1) + j & \text{for } j \% 2 = 1 \text{ and for } \forall i \\
  d \cdot (i-1) + j & \text{for } i \% 2 = 0 \\
  d \cdot i + j & \text{for } i \% 2 = 1 
\end{cases} \]  

Initialize Ncbps and d

for j=0 to d-1, j++

for i=0 to (Ncbps/d)-1, i++

if (j mod 2 =0)

\[ k_n = d \cdot i + j \]

else

if (i mod 2 =0)

\[ k_n = d \cdot (i+1) + j \]

else

\[ k_n = d \cdot (i-1) + j \]

end if

end if

end for

end for
Algorithm for 64-QAM

\[ k_{n,64-QAM} = \begin{cases} 
  d \cdot i + j & \text{for } j \mod 3 = 0 \text{ and for } \forall i \\
  d \cdot (i - 2) + j & \text{for } j \mod 3 = 1 \text{ and for } i \mod 3 = 2 \\
  d \cdot (i + 1) + j & \text{for } j \mod 3 = 1 \text{ and for } i \mod 3 \neq 2 \\
  d \cdot (i + 2) + j & \text{for } j \mod 3 = 2 \text{ and for } i \mod 3 = 0 \\
  d \cdot (i - 1) + j & \text{for } j \mod 3 = 2 \text{ and for } i \mod 3 \neq 0 
\end{cases} \] (7)

Initialize Ncbps and d

for j=0 to d-1, j++
for i=0 to (Ncbps/d)-1, i++
  if (j mod 3 =0)
    kn=d*i+j
  elseif (j mod 3 =1)
    if (i mod 3 =2)
      kn=d*(i-2)+j
    else
      kn=d*(i+1)+j
    end if
  else
    if (i mod 3 =0)
      kn=d*(i+2)+j
    else
      kn=d*(i-1)+j
    end if
  end if
end for
end for

IV. PROPOSED WORK

In the base paper, the authors have used multiplexer with the select lines to select the sequential address fed with the code rates as shown in the fig below, we can observe that the starting address fed starts from 5 then 8 then 11 and so on... these are the bits obtained depending upon the interleaver depths that is 96/16=6 so 0 to 5 hence the number 5 is chosen for first input, then for the depth 144, 144/16=9 so 0 to 8 hence number 8 is chosen for second input and similarly it goes on until the final depth is reached. It can be more clearly understood by viewing to the table shown below.

<table>
<thead>
<tr>
<th>Ndepth</th>
<th>No.of Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>96</td>
<td>6</td>
</tr>
<tr>
<td>144</td>
<td>9</td>
</tr>
<tr>
<td>192</td>
<td>12</td>
</tr>
<tr>
<td>288</td>
<td>18</td>
</tr>
<tr>
<td>384</td>
<td>24</td>
</tr>
<tr>
<td>432</td>
<td>27</td>
</tr>
<tr>
<td>480</td>
<td>30</td>
</tr>
<tr>
<td>576</td>
<td>36</td>
</tr>
</tbody>
</table>
We don’t have any problem with these inputs but using multiplexer with so much of select lines results in complexity of hardware, power and huge resource. Instead, in my paper I have tried with divider circuit such that whenever the depth is applied as input to the block, it itself inbuilt calculates the address such that no manipulation is required manually outside the block hence the resource, power and complexity can be significantly reduced by the considerable amount. The fig intended for my proposed work is as shown in fig below, the common modules share between these three modulation schemes are, comparator, adder, multiplexer and counters. The modulo and subtractor are the extra blocks used in QAM.

![Fig 4: QPSK block with manual computation.](image)

![Fig 5: QPSK without manual computation.](image)

Here if Ncbps is given as input, the total number of columns can be computed as Ncbps/d. Therefore the column number varies from 0 to (Ncbps/d)-1. This can be obtained by using divider and subtractor circuit that is instead of using 8*1 multiplexer, the input Ncbps is given to the divider circuitry and the output obtained from this is subtracted by 1. These two operations are done by divider circuitry. Similarly modifications are showed for 16-QAM and 64-QAM respectively.

![Fig 6: 16-QAM with manual computation of column number.](image)
Relation between deinterleaver depth and total number of columns

<table>
<thead>
<tr>
<th>$N_{\text{chps}}$</th>
<th>No. of Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>192</td>
<td>12</td>
</tr>
<tr>
<td>288</td>
<td>18</td>
</tr>
<tr>
<td>384</td>
<td>24</td>
</tr>
<tr>
<td>576</td>
<td>36</td>
</tr>
</tbody>
</table>

Fig 7: 16-QAM without manual computation of column number.

Relation between deinterleaver depth and total number of columns

<table>
<thead>
<tr>
<th>$N_{\text{chps}}$</th>
<th>No. of Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>288</td>
<td>18</td>
</tr>
<tr>
<td>384</td>
<td>24</td>
</tr>
<tr>
<td>432</td>
<td>27</td>
</tr>
<tr>
<td>576</td>
<td>36</td>
</tr>
</tbody>
</table>

Fig 8: 64-QAM with manual computation of column numbers.
Fig 9: 64-QAM without manual computation of column numbers.

Another fig below shows that how all three modulation schemes are embedded after sharing common resource between them.

Fig 10: Top level view of complete deinterleaver address generator

V. SIMULATION RESULTS

Simulation results are obtained fig 11-16 using Xilinx Isim for all permissible modulation types and code rates. Results are shown for selected data rate of each modulation type as shown below, now after the simulation is viewed, the output can be check or verify using the Spartan-3 kit of Field Programmable Gate Array (FPGA)

Fig 11: Simulation results for QPSK with Ncbps=96
Fig 12: QPSK with Ncbps=288.

Fig 13: QPSK with Ncbps=576.

Fig 14: 16-QAM with Ncbps=576.

Fig 15: 64-QAM with Ncbps=288.

Fig 16: 64-QAM with Ncbps=576.
VI. COMPARISON OF RESULTS

Table below shows the comparison obtained between the base paper and our proposed work.

<table>
<thead>
<tr>
<th>FPGA Parameters</th>
<th>Proposed technique</th>
<th>Base Paper</th>
<th>LUT based technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slices</td>
<td>1%</td>
<td>3.49%</td>
<td>17.66%</td>
</tr>
<tr>
<td>Flip Flops</td>
<td>0%</td>
<td>0.50%</td>
<td>0.78%</td>
</tr>
<tr>
<td>4 input LUTs</td>
<td>1%</td>
<td>3.35%</td>
<td>17.15%</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>132.39MHz</td>
<td>121.82MHz</td>
<td>62.51MHz</td>
</tr>
</tbody>
</table>

VII. CONCLUSION

Area and delay efficient circuitry of address generator for WiMAX 2-D Deinterleaver without using floor function for all permissible modulation schemes and code rates are implemented using Xilinx Field Programmable Gate Array (FPGA) without manual computation for total number of columns. Hence by this, a simple, novel, efficient resource utilization with low power technique is been achieved with satisfactory results.

REFERENCES


