International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:3, No:4, 2009

Low Complexity Multi Mode Interleaver Core for WiMAX with Support for Convolutional Interleaving

Rizwan Asghar and Dake Liu

***Abstract*—**A hardware efficient, multi mode, re-configurable architecture of interleaver/de-interleaver for multiple standards, like DVB, WiMAX and WLAN is presented. The interleavers consume a large part of silicon area when implemented by using conventional methods as they use memories to store permutation patterns. In addition, different types of interleavers in different standards cannot share the hardware due to different construction methodologies. The novelty of the work presented in this paper is threefold: 1) Mapping of vital types of interleavers including convolutional interleaver onto a single architecture with flexibility to change interleaver size; 2) Hardware complexity for channel interleaving in WiMAX is reduced by using 2-D realization of the interleaver functions; and 3) Silicon cost overheads reduced by avoiding the use of small memories. The proposed architecture consumes 0.18mm^2 silicon area for 0.12µm process and can operate at a frequency of 140 MHz. The reduced complexity helps in minimizing the memory utilization, and at the same time provides strong support to on-the-fly computation of permutation patterns.

***Keywords*—**Hardware interleaver implementation, WiMAX, DVB, block interleaver, convolutional interleaver, hardware multiplexing.

I. Introduction

THE focus of this research is to enable the hardware re-use for FEC subsystems. Among FEC subsystems, interleavers and de-interleavers appeared to be the most silicon consuming. This is due to the silicon cost of the addressing/permutation tables used in the conventional approaches. Due to rapidly changing technology trends, many consumer products require the adaptation of different interleaving standards. Therefore a re-configurable architecture supporting multiple radio communication standards with minimal hardware cost is always beneficial. This paper presents a flexible and low cost hardware interleaver architecture, which covers the block interleavers for channel interleaving and duo-binary turbo codes adopted in IEEE 802.16e [1] and the convolutional interleaver as specified in ETSI EN 300-744 [2].

System level overview for IEEE 802.16e called WiMAX is shown in Fig. 1(a) and for ETSI EN 300-744 called DVB is shown in Fig. 1(b). WiMAX uses the block interleaver for channel interleaving and duo-binary turbo code interleaving.

R. Asghar is with Department of Electrical Engineering, Linköping University, SE-58183, Linköping, Sweden (phone: +46 (0)13 28 2313; fax: +46 (0)13139 282 ; e-mail: rizwan@ isy.liu.se).

D. Liu is with Department. of Electrical Engineering, Linköping University, SE-58183, Linköping, Sweden (e-mail: dake@ isy.liu.se).

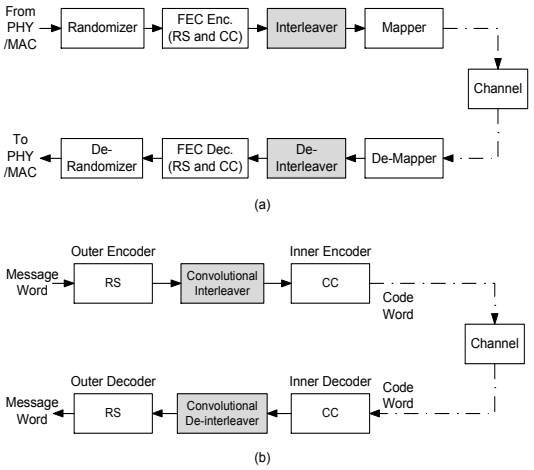

Fig. 1. Overview of encoding in (a) WiMAX channel, (b) DVB channel.

In block interleavers, the data is written sequentially in a memory and read in a random order after applying certain permutations. The block interleaver can also be considered as a row-column matrix. In this case, data is written row-wise in a memory configured as a row-column matrix and then read column-wise after applying certain intra-row and inter-row permutations. On the other hand, the convolutional interleavers use multiple first-in-first-out (FIFO) cells with different widths and depths to disperse the adjacent information. In addition to different structures of block interleaving and convolutional interleaving, they also impose different latency measures. All these mismatches between different types of interleavers make it hard to implement them on a single architecture. Recently [4] [5] has proposed an architecture to implement the de-interleaver for WiMAX and DVB in a single chip using 0.6 mm^2 and 0.484 mm^2 area respectively. A fast interleaver design covering 802.16 and 802.11 has been proposed in [6] with some extra hardware cost. An FPGA implementation of DVB interleaver is described in [7] with emphasis on minimizing memory utilization. The architecture given in [4] [5] and [7] use six small memories each having size of 255 byte or less, which turns out to be hardware inefficient due to significant overheads of small memories. Some commercial implementations for interleaver/de-interleavers are also available from major FPGA vendors like Xilinx, Altera and Lattice Semiconductor. The architecture insights for these implementations are not disclosed and they are provided as IP Blocks. However, the available literature

[18]–[20] explains the memory configuration, performance and area utilization in terms of logic elements (LEs) or lookup-tables (LUTs).

Our proposed architecture uses two single port memories of size 512 bytes and 1024 bytes, thus minimizing the overheads due to small memories. It enables the hardware re-use for different types of interleavers to compute the interleaved addresses for writing/reading the data to/from memory, in order to achieve the low cost solution. Low cost solution is also supported by realizing the 1-dimensional permutation functions provided for WiMAX channel interleaver into 2-dimentional functions, where it is easy to realize the interleaver as a row-column matrix. Section 2 of this paper presents the 2-D transformation steps for WiMAX/WLAN channel interleaver and section 3 describes the interleaver construction for duo-binary turbo codes. The convolutional interleaver construction is provided in section 4 while section 5 presents the complete hardware solution for re-configurable interleaver. Section 6 and 7 provide the implementation results and conclusion respectively.

II. WiMAX/WLAN Channel Interleaver

WiMAX uses Read-Solomon and convolutional encoding followed by an interleaver as shown in Fig. 1(a) to detect and correct errors to improve the performance of the communication system. Different interleaving patterns apply for different modulation schemes BPSK/QPSK, 16-QAM and 64-QAM. The channel interleaving in WiMAX/WLAN [3] is based on a block interleaver, which is expressed in the form of a set of two equations for two steps of permutations. The first step ensures that adjacent coded bits are mapped onto non-adjacent subcarriers, while the second step ensures that adjacent coded bits are mapped alternately onto less or more significant bits of constellation, thus avoiding long runs of lowly reliable bits.

The first permutation m_k for index k is defined by:

$$m_k = \left(\frac{N_{cbps}}{d}\right)\cdot\left(k\,\%\,d\right)+\left\lfloor\frac{k}{d}\right\rfloor \qquad (1)$$

Here N_{cbps} is the block size corresponding to number of coded bits per allocated sub-channels per OFDM and typical value for d used in WiMAX is 12 and 16. The operator % is defined as the modulo function computing the remainder and the operator $\lfloor x\rfloor$ is the floor function i.e. rounding towards zero. The second permutation j_k for index k is given by:

$$j_k = s\cdot\left\lfloor\frac{m_k}{s}\right\rfloor+\left(\left(m_k+N_{cbps}-\left\lfloor d\cdot\frac{m_k}{N_{cbps}}\right\rfloor\right)\%\,s\right) \qquad (2)$$

The parameter s is defined as $s = ceil\left(N_{cpc}/2\right)$, where N_{cpc} is number of coded bits per sub-carrier, i.e., 1, 2, 4 or 6 for BPSK, QPSK, 16-QAM or 64-QAM respectively and $ceil$ operation is rounding towards infinity. The de-interleaver, which performs the inverse operation, is also defined by the two permutations. Let n be the index of received bits within the received block of N_{cbps} bits. The first permutation m_n for index n is defined by:

$$m_n = s\cdot\left\lfloor\frac{n}{s}\right\rfloor+\left(\left(n+\left\lfloor d\cdot\frac{n}{N_{cbps}}\right\rfloor\right)\%\,s\right) \qquad (3)$$

The second permutation k_n for index n is given by:

$$k_n = d\cdot m_n-\left(\left(N_{cbps}-1\right)\cdot\left\lfloor d\cdot\frac{m_n}{N_{cbps}}\right\rfloor\right) \qquad (4)$$

The range of n and k for eq. (1) to (4) is defined as 0, 1, 2, ... $(N_{cpbs}-1)$. If we try to implement the two steps of permutations by direct computation then they are found to be quite hardware inefficient. This is due to the presence of complex functions like floor function and modulo function.

The alternate is to consider the two steps as one step and find the correlation between input and output which should be hardware efficient. We present here the idea of realizing the one dimensional equations into a joint 2-dimensional expression. It is not necessary to transform both set of equations to 2-D space and implement separately, as they are inverse of each other. Thus only one set of equations can be transformed for efficient hardware implementation and same can be used for other by just swapping the order of read and write of data into memory. The following sub-sections present the transformation steps for all kinds of modulation schemes used in WiMAX/WLAN.

A. BPSK / QPSK

Due to $ceil$ operation the parameter s is 1 for both BPSK and QPSK. Defining $N = N_{cbps}$ eq. (3) simplifies to $m_n = n + 0 = n$, and therefore eq. (4) becomes:

$$k_n = d\cdot n-\left(\left(N-1\right)\cdot\left\lfloor\frac{d\cdot n}{N}\right\rfloor\right)$$

$$k_n = d\cdot\left(n-\frac{N}{d}\cdot\left\lfloor\frac{d\cdot n}{N}\right\rfloor\right)+\left\lfloor\frac{d\cdot n}{N}\right\rfloor$$

$$k_n = d\cdot\beta_n+\gamma_n \qquad (5)$$

Where

$$\beta_n = n-\frac{N}{d}\cdot\left\lfloor\frac{d\cdot n}{N}\right\rfloor \text{ and } \gamma_n = \left\lfloor\frac{d\cdot n}{N}\right\rfloor = \left\lfloor\frac{n}{N/d}\right\rfloor$$

Due to the presence of floor function, it is difficult to work out a complete algebraic solution for these equations, however looking at the behavior of different terms and verifying for all possible block sizes, we try to re-structure the equations. MATLAB is used for verification of new structures at all stages. For a simple illustration, an example case of BPSK with 2 sub channels and $d = 16, N = 32$, is taken and behavior of β_n is analyzed against the index n.

$$\beta_n = n-2\cdot\left\lfloor\frac{n}{2}\right\rfloor$$

$$n=0 \;\rightarrow\; \beta_n=0 \;\rightarrow\; \beta_n=(0\%2)$$
$$n=1 \;\rightarrow\; \beta_n=1 \;\rightarrow\; \beta_n=(1\%2)$$
$$n=3 \;\rightarrow\; \beta_n=0 \;\rightarrow\; \beta_n=(2\%2)$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$n=n \;\rightarrow\; \beta_n=1 \;\rightarrow\; \beta_n=(n\%2)$$

After checking all cases for BPSK and QPSK (i.e. sub-channels 1,2,4,8,16), β_n can be generalized as:

$$\beta_n=\left(n\%\frac{N}{d}\right)$$

Thus for BPSK or QPSK case, eq. (5) can now be written as :

$$k_n=d\cdot\left(n\%\frac{N}{d}\right)+\left\lfloor\frac{d\cdot n}{N}\right\rfloor \qquad (6)$$

Introducing 2 dimensions i and j (i.e. a two dimensional array), for which j increments when i expires, the ranges for i and j can easily be selected as mentioned below:

$$i=0,1,\ldots\ldots\left(\frac{N}{d}-1\right)\ which\ satisfies\ against\ 'n'\ if\ \ i=\left(n\%\frac{N}{d}\right) \qquad (7)$$

$$j=0,1,\ldots\ldots(d-1)\ \ with\ behavior\ against\ 'n'\ \ j=\left\lfloor\frac{n}{N/d}\right\rfloor \qquad (8)$$

The interleaver can now be realized as a 2D row-column matrix with size $i\times j$. Total number of columns is d, defined by the limit on j and total number of rows is N/d. Eq. (6) can be written in the form:

$$k_n \;\equiv\; k_{i,\,j}=d\cdot i+j \qquad (9)$$

Here i and j are row and column counters respectively but at the same time, they also provide the inter-row and inter-column permutations. The case of BPSK and QPSK is the simplest one as it does not carry any specific inter-row or inter-column permutation pattern due to the parameter $s=1$. That is why we end up with a relatively simple hardware needing just one addition and a multiplication as shown in Fig. 2(a), but it provides the basis for analysis for 16-QAM and 64-QAM which are more complicated.

B. 16-QAM

The parameter s is 2 for 16-QAM therefore eq. (3) and eq. (4) can be written as:

$$m_n=2\cdot\left\lfloor\frac{n}{2}\right\rfloor+\left(\left(n+\left\lfloor\frac{d\cdot n}{N}\right\rfloor\right)\%\ 2\right) \qquad (10)$$

$$k_n=d\cdot\left(m_n-\frac{N}{d}\cdot\left\lfloor\frac{d\cdot m_n}{N}\right\rfloor\right)+\left\lfloor\frac{d\cdot m_n}{N}\right\rfloor \qquad (11)$$

Two terms can again be defined as β_n and γ_n .

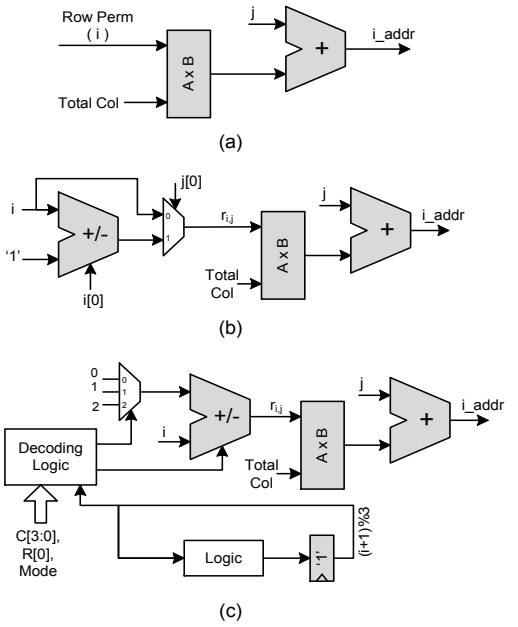

Fig. 2. HW realization for channel interleaving in WiMAX (a) BPSK-QPSK, (b) 16-QAM, (c) 64-QAM.

0	17	2	....	29	14	31
16	1	18	....	13	30	15
32	49	34	....	61	46	63
48	33	50	....	45	62	47

(a)

0	17	34	....	29	46	15
16	33	2	....	45	14	31
32	1	18	....	13	30	47
48	65	82	....	77	94	63
64	81	50	....	93	62	79
80	49	66	....	61	78	95

(b)

Fig. 3. Examples of data interleaving for (a) 16-QAM, N=64; (b) 64-QAM, N=96.

$$\beta_n=m_n-\frac{N}{d}\cdot\left\lfloor\frac{d\cdot m_n}{N}\right\rfloor \ \text{ and } \ \gamma_n=\left\lfloor\frac{d.m_n}{N}\right\rfloor \qquad (12)$$

Therefore $\qquad k_n=d\cdot\beta_n+\gamma_n$

After verifying for all the range for WiMAX, the parameter γ_n can be written as:

$$\gamma_n=\left\lfloor\frac{d.m_n}{N}\right\rfloor=\left\lfloor\frac{d.n}{N}\right\rfloor=\left\lfloor\frac{n}{N/d}\right\rfloor=j \qquad (13)$$

However, it does not mean that m_n is equal to n all the time. This is valid only due to the presence of floor function around it. Using definitions in eq. (10) and eq. (13), β_n can be re-written as:

$$\beta_n=2\cdot\left\lfloor\frac{n}{2}\right\rfloor+\left(\left(n+\left\lfloor\frac{d\cdot n}{N}\right\rfloor\right)\%\ 2\right)-\frac{N}{d}\cdot\left\lfloor\frac{d\cdot n}{N}\right\rfloor \qquad (14)$$

Now we try to re-arrange this equation to find some new structure which is similar to eq. (9). For illustration purposes some steps for the 16-QAM example case with $d=16$ and $N=64$ are given below:

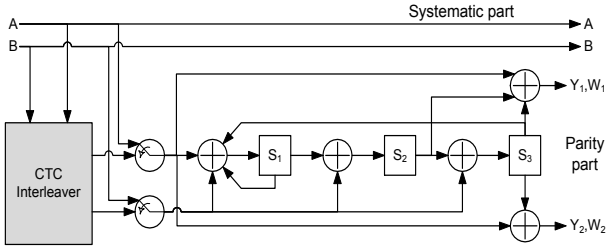

Fig. 4. CTC Encoder with Interleaver.

$$
\begin{aligned}
&n=0 \rightarrow \beta_n=0 \rightarrow \beta_n=0\left[1-(0\,\%\,2)\right]+\\
&\left\{(0+1)\left[1-(0\,\%\,2)\right]+(0-1)(0\,\%\,2)\right\}(0\,\%\,2)\\
&n=1 \rightarrow \beta_n=1 \rightarrow \beta_n=1\left[1-(0\,\%\,2)\right]+\\
&\left\{(1+1)\left[1-(1\,\%\,2)\right]+(1-1)(1\,\%\,2)\right\}(0\,\%\,2)\\
&n=2 \rightarrow \beta_n=2 \rightarrow \beta_n=2\left[1-(0\,\%\,2)\right]+\\
&\left\{(2+1)\left[1-(2\,\%\,2)\right]+(2-1)(2\,\%\,2)\right\}(0\,\%\,2)\\
&n=3 \rightarrow \beta_n=3 \rightarrow \beta_n=3\left[1-(0\,\%\,2)\right]+\\
&\left\{(3+1)\left[1-(3\,\%\,2)\right]+(3-1)(3\,\%\,2)\right\}(0\,\%\,2)\\
&n=4 \rightarrow \beta_n=1 \rightarrow \beta_n=0\left[1-(1\,\%\,2)\right]+\\
&\left\{(0+1)\left[1-(0\,\%\,2)\right]+(0-1)(0\,\%\,2)\right\}(1\,\%\,2)\\
&n=5 \rightarrow \beta_n=1 \rightarrow \beta_n=1\left[1-(1\,\%\,2)\right]+\\
&\left\{(1+1)\left[1-(1\,\%\,2)\right]+(1-1)(1\,\%\,2)\right\}(1\,\%\,2)\\
&\qquad\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&\qquad\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&\beta_n \equiv r_{i,j}=\left[1-(j\,\%\,2)\right]i+\left[(j\,\%\,2)\right]\times\\
&\qquad\left\{(i+1)\left[1-(i\,\%\,2)\right]+(i-1)(i\,\%\,2)\right\}
\end{aligned}
\tag{15}
$$

Where i and j are defined with ranges as mentioned in eq. (7) and eq. (8) i.e.

$$i=\left(n\,\%\,\frac{N}{d}\right)\quad and\quad j=\left\lfloor\frac{n}{N/d}\right\rfloor$$

Verifying for all the cases in 16-QAM, we reach to a new structure for β_n as given in eq. (15). This structure is not as simple as that of BPSK/QPSK case. The reason is the presence of permutation pattern in 16-QAM case. Considering the 2 dimensions i and j, the 2D transformation of interleaver for 16-QAM can be described as:

$$k_n \equiv k_{i,j}=d\cdot r_{i,j}+j \tag{16}$$

The parameter $r_{i,j}$ provides an intra-row permutation pattern sequence for selective columns, such that a permutation is applied for all alternate columns $(2y+1)^{th}$ and no permutation is applied for each $2y^{th}$ columns, where $y=1,2,\ldots\ldots\ldots d/2$. Considering total number of rows as R, the required inter-row permutation for row number i $(0,1,2\ldots\ldots\ldots R-1)$ is $i+1$ and $i-1$ for each $2i^{th}$ and $(2i+1)^{th}$ row respectively. Looking at eq. (16), the generic structure for 16-QAM is same as that of eq. (9) except the additional complexity for selective row permutation. The structure of eq. (16) is easy to implement with a row and column counter i and j. The terms with modulo function can be controlled by just the LSB of corresponding variable and the rest can be managed by a lookup table (LUT) or an adder. As number of rows in the block can be many (upto 96 for WiMAX) thus use of LUT is not efficient here. Instead we can use a 7 bit adder, which can also give the benefit of generalizing the implementation. The hardware realization for interleaver address generation for 16-QAM in WiMAX is shown in Fig. 2(b).

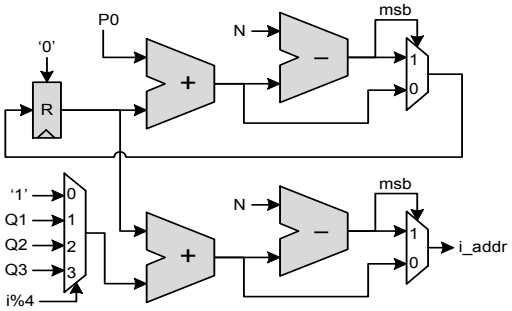

Fig. 5. Hardware for CTC Interleaver.

C. 64-QAM

As number of coded bits per sub-carrier are 6 for 64-QAM transmission, thus using the parameter $s=3$, eq. (3) is written as:

$$m_n=3\cdot\left\lfloor\frac{n}{3}\right\rfloor+\left(\left(n+\left\lfloor\frac{d\cdot n}{N}\right\rfloor\right)\%\ 3\right) \tag{17}$$

Defining the two terms β_n and γ_n as given in eq. (12) and eq. (13) we can write expression for β_n for 64-QAM as:

$$\beta_n=3\cdot\left\lfloor\frac{n}{3}\right\rfloor+\left(\left(n+\left\lfloor\frac{d\cdot n}{N}\right\rfloor\right)\%\ 3\right)-\frac{N}{d}\cdot\left\lfloor\frac{d\cdot n}{N}\right\rfloor \tag{18}$$

Again applying the same re-structuring exercise as we did for 16-QAM case, we reach to an even more complicated 2D-structure for β_n. Due to increased complexity for permutation patterns for 64-QAM the intermediate steps carry much longer terms, thus we directly present the final structure for β_n.

$$
\begin{aligned}
\beta_n \Rightarrow r_{i,j}=&\left[(1-j')+\frac{j'(j'-1)}{2}\right]\cdot i\\
&+\left\{[j'-(j'-1)]\left\{\begin{matrix}(i-2)\left[(1-i')+\dfrac{i'(i'-1)}{2}\right]\\ +(i+1)\left[i'-\dfrac{i'(i'-1)}{2}\right]\end{matrix}\right\}\right\}\\
&+\left\{\frac{j'(j'-1)}{2}\left\{\begin{matrix}(i+2)[i'-i'(i'-1)]\\ +(i-1)[(1-i')+i'(i'-1)]\end{matrix}\right\}\right\}
\end{aligned}
\tag{19}
$$

Here i and j are row and column count respectively, with the ranges mentioned in eq. (7) and eq.(8). The new parameters i' and j' are defined as below:

$$i' = (i+1) \ \% \ 3 \quad \text{and} \quad j' = j \ \% \ 3$$

The term β_n for 64-QAM provides the selective inter-row permutation for every $(3j+1)^{th}$ and $(3j+2)^{th}$ column. The permutation for all these columns is within 3 rows and afterwards it is repeated. Considering total number of rows as R, the inter-row permutation in $(3j+1)^{th}$ columns for row number i $(0,1,2 \ldots\ldots\ldots R-1)$ is $i+1$, $i+1$, $i-2$ for $3i^{th}$, $(3i+1)^{th}$ and $(3i+2)^{th}$ row respectively. The inter-row permutation for $(3j+2)^{th}$ columns is $i+2$, $i-1$ and $i-1$ for $3i^{th}$, $(3i+1)^{th}$ and $(3i+2)^{th}$ row respectively. Examples of address permutations for 16-QAM and 64-QAM with small block sizes are shown in Fig. 3, which also correspond to the permutation patterns described here.

Combining the interleaver structure for all the cases, the 2D single step generic interleaver function $k_{i,j}$ can be described as:

$$k_{i,j} = d \cdot r_{i,j} + j \qquad (20)$$

Where $r_{i,j} = i$ for BPSK/QPSK and it is defined by eq. (15) and eq. (19) for 16-QAM and 64-QAM respectively. Although eq. (19) looks very long and complicated, but eventually, we get a hardware efficient solution. Additionally we stick to the generic interleaver hardware for all types of modulation schemes. The implementation of modulo terms j%3 and $(i+1)$%3 and some other terms inside braces are easier to generate through a very small lookup table. Other permutation values with addition and subtraction can be implemented with the help of a multiplexer and an adder. The hardware realization for 64-QAM interleaver is shown in Fig. 2(c).

III. Interleaver for Duo-Binary Turbo Codes

The turbo codes [10] invented in 1993 captured great importance due to exhibiting near Shannon-limit performance. Recently, double binary turbo codes (convolutional turbo codes, CTC) have received a great attention as they are adopted in several mobile radio systems such as DVB and WiMAX. They can offer many advantages like performance, over the classical single-binary turbo codes [11]. Fig. 4 shows the block diagram for the duo-binary encoder including an interleaver. In CTC the information is treated as pair of bits and the two output parts, systematic output and parity output are almost un-correlated due to the presence of interleaver. The interleaver for CTC is a two step interleaver and is defined for a particular block size N. Parameters for block size, modulation scheme and coding rate are provided in WiMAX standard [1], and are designated as P_0, P_1, P_2 and P_3. Two steps of interleaving are described below:

Step 1:
Let the incoming sequence be
$u_0 = [(A_0, B_0), (A_1, B_1), (A_2, B_2), \ldots. (A_{N-1}, B_{N-1})]$,
for $i = 0 \ldots\ldots N-1$,
if $(i\%2) = 1$ *then* $(A_i, B_i) = (B_i, A_i)$.

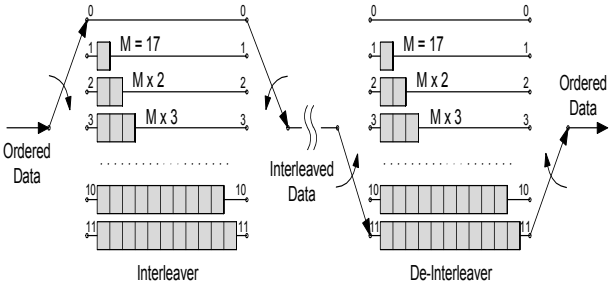

Fig. 6. Convolutional interleaver and de-interleaver in DVB.

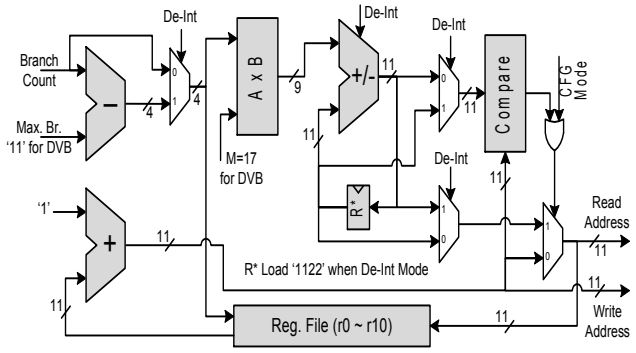

Fig. 7. Hardware for RAM read/write address generation for convolutional (de) interleaver in DVB.

The new sequence is
$u_1 = [(A_0, B_0), (B_1, A_1), (A_3, B_3), \ldots. (B_{N-1}, A_{N-1})]$

Step 2:
The function $P(j)$ *provides the address of the couple from the sequence* u_1 *that shall be mapped onto address* j *of the interleaved sequence.* $P(j)$ *is defined by the set of four expressions with a switch selection as follows:*
for $j = 0 \ldots\ldots N-1$,
switch $(j \ \% \ 4)$*:*
case 0: $P(j) = (P_0 \ . \ j + 1) \ \% \ N$
case 1: $P(j) = \left(P_0 \ . \ j + 1 + \frac{N}{2} + P1\right) \ \% \ N$
case 2: $P(j) = (P_0 \ . \ j + 1 + P1) \ \% \ N$
case 3: $P(j) = \left(P_0 \ . \ j + 1 + \frac{N}{2} + P3\right) \ \% \ N$

The four equations given in step 2 can be written in combined form as:

$$P(j) = \left(P_0 . j + Q_j\right) \% N \qquad (21)$$

Where

$$Q_j = \begin{cases} 1 & ; \ if\ (j\%4 = 0) \\ 1 + N/2 + P_1 & ; \ if\ (j\%4 = 1) \\ 1 + P_2 & ; \ if\ (j\%4 = 2) \\ 1 + N/2 + P_3 & ; \ if\ (j\%4 = 3) \end{cases}$$

Let

$$\beta_j = P_0 . j \ \% \ N \quad \rightarrow \quad \beta_0 = 0$$

Then recursively:

$$\beta_{j+1} = \left(\beta_j + P_0\right) \% \ N$$

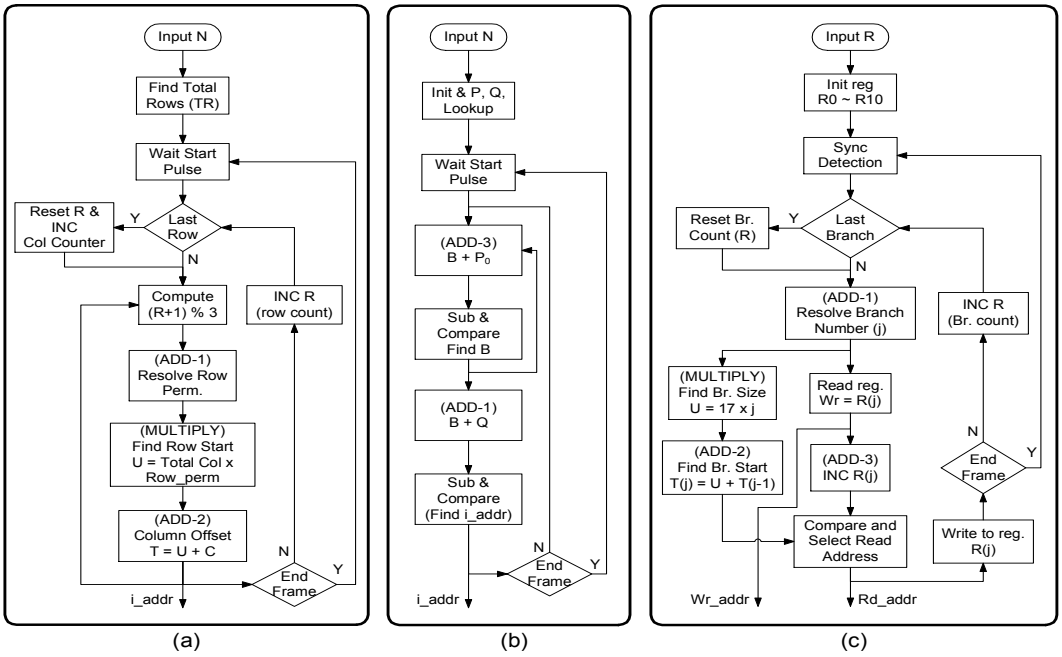

Fig. 8. Flow Graph for (a) Channel Interleaving in WiMAX, (b) CTC Interleaving, (c) The read/write address computation for DVB.

Now if we know β_j by recursive computation, $P(j)$ can be computed from the following equation.

$$P(j) = \left(\beta_j + Q_j\right)\%\, N \tag{22}$$

By looking at the range of parameters β_j and Q_j, their sum cannot be larger than $2N$. Thus $P(j)$ can be computed by using addition and subtraction with compare and select logic, as shown in Fig. 5 where the values for Q_j are provided through a lookup table. Same type of hardware is used in the recent work [12] and [13] to implement the interleaver for complete CTC decoder design. Our objective is to integrate this hardware with other interleaver structures in a multiplexed way to achieve a flexible and reconfigurable interleaver which can support the WiMAX standard, in case the turbo code is used.

IV. Convolutional Interleaver for DVB

The convolutional interleaver used in DVB is based on the Forney approach [8] which is compatible with Ramsey type III approach [9]. In order to distribute burst errors, which are not corrected by Viterbi decoder in receiver, a convolutional interleaver is used in transmitter between RS encoding and convolutional encoding as shown in Fig. 1(b). Thus a de-interleaver has to be incorporated in the receiver before the RS-decoder, to be able to decode the packets. The convolutional interleaver for DVB consist of $I = 12$ branches and each branch j is composed of first-in-first-out (FIFO) shift registers with depth $j \times M$, where $M = 17$ for DVB. The packet of 204 bytes consisting of one sync byte $(0 \times 47\ or\ 0 \times B8)$ is entered into the interleaver in a periodic way. For synchronization purpose the sync bytes are always routed to $branch\ 0$ of interleaver as shown in

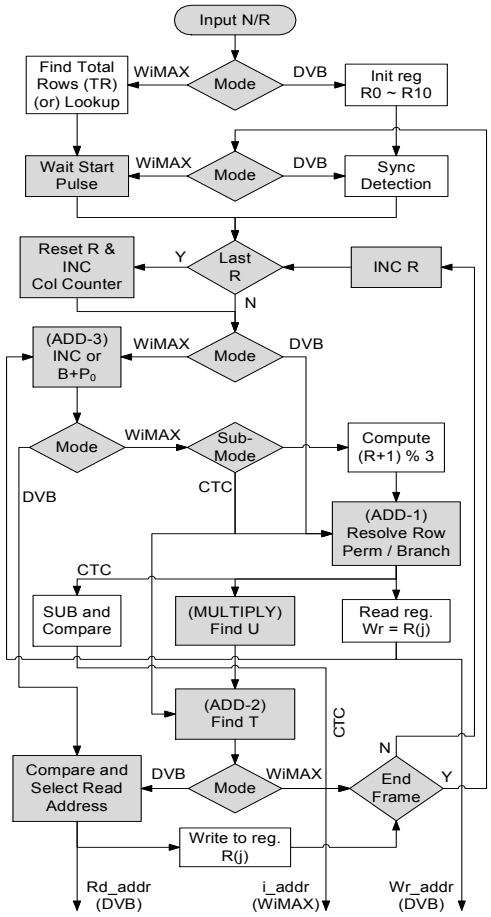

Fig. 9. Flow Graph for combined interleaver (Gray blocks show the flow overlap and hardware sharing between different interleavers).

Fig. 6. The convolutional interleaving provides half the latency as compared to block interleaving and also consumes less memory size. Recently convolutional interleavers have been analyzed to work with Turbo Codes [14] – [17], which make them more versatile, thus general and re-configurable convolutional interleaver architecture can be of significance. The improvement in performance of turbo codes using convolutional interleavers provides the motivation to integrate the functionality of convolutional interleaver with block interleavers to increase the flexibility for working with turbo codes. This section presents the hardware implementation of convolutional interleaver which will be integrated with block interleavers in next section.

Due to large consumption of silicon area, the implementation of the convolutional interleaver or de-interleaver using first-in-first-out (FIFO) register cells would be very hardware inefficient. To achieve a hardware efficient solution, RAM based implementation is proposed. The memory partitioning is made in such a way that by applying appropriate read/write addresses in a cyclic way, it exhibits the branch behavior as required by convolutional interleaver. RAM write and read addresses are generated by the hardware shown in Fig. 7. The hardware components used here are almost the same as used by interleaver design for WiMAX, thus providing the basis for multiplexing the hardware blocks for re-use. The main difference is the use of 11 registers to keep track of next write addresses for each branch, which is the idea of using cyclic pointers instead of using FIFO shift registers. For each branch the corresponding write address is provided by the concerned pointer register and next write address (which is also called current read address) is computed by using an addition and a comparison with the branch boundaries. The branch boundaries are computed on the fly using an adder and a multiplier in connection with a branch counter.

For implementing convolutional de-interleaver same hardware is used by implementing the branch counter in reverse order (decrementing by 1). In this way same branch boundaries are used, and the only difference is that the sync byte in the data is now synchronized with the largest branch size as shown in Fig. 6. Keeping the same branch boundaries for de-interleaver, the width of pointer register becomes fixed. This gives an additional benefit that the width of pointer register may be optimized efficiently, i.e. instead of using all pointer register of width 11 bit we can use smaller width for the lower branches and larger width for the upper branches.

V. Complete Hardware

Fig. 8 presents the flow graphs for the computation of interleaved addresses for memory read and write for the interleaver types covered in sections 2 – 4. The combined flow graph illustrating flow sharing between different implementations is presented in Fig. 9. In order to fulfill the shared flow, in a multiplexed way, the complete hardware for the data interleaving or de-interleaving for multiple standards is divided into sub-blocks like control FSM block, address generation block and memory organization block. These blocks are briefly described in the following sub-sections.

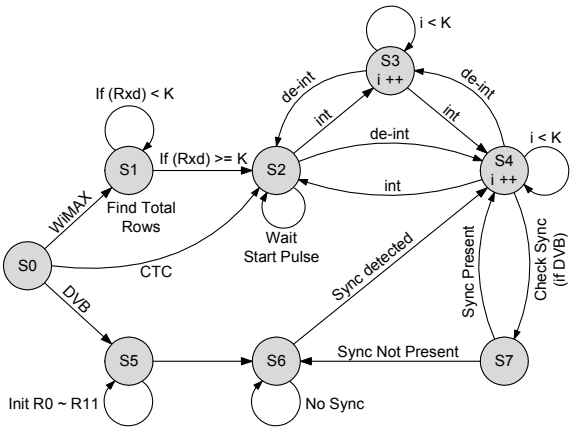

Fig. 10. Control FSM for combined hardware interleaver.

A. Control FSM

An eight state control FSM shown in Fig. 10 is used to synchronize the flow for address computation. The control FSM serves different initialization requirements for different standards at startup. In the initialization phase for WiMAX/WLAN, the controller computes the number of rows for a particular block size in state S1, while for DVB the pointer registers are initialized in state S5 to their respective start points. States S2 – S4 are shared for channel interleaving and duo-binary turbo code interleaving, whereas state S2 also serves for synchronization with the external world. After initialization, the FSM keeps track of block size by employing row and column count, thus providing the block synchronization required for each standard. State S6 and S7 are used for sync detection in case of DVB interleaver. Once sync pattern is detected, state S4 is again used for branch counting and branch synchronization.

B. Address Computation Circuitry

The address computation is achieved for different standards by multiplexing the hardware covered in sections 2 – 4. Some additional multiplexers and glue logic is used to support the re-configurability for different standards and different modulation schemes. The hardware multiplexed circuit for combined address computation is shown in Fig. 11(a). Here, the multiplier can further be optimized to only one adder, but it is kept there to make the design general for any branch size and any number of columns in the block interleaver. The address computation circuitry also involves a lookup table implementing the decoding logic for operand selection and a register file consisting of 12 pointer registers. The pointer registers are mainly used for DVB, thus not sharing with other types of implementations. The computation intensive blocks in the address computation circuitry are 4 adders/subtractors, a multiplier and a comparator. Except one subtractor which is only used for CTC interleaver, all the rest of the blocks are being shared by different interleaver implementations. This provides a highly multiplexed architecture with good hardware utilization for different interleaver implementations.

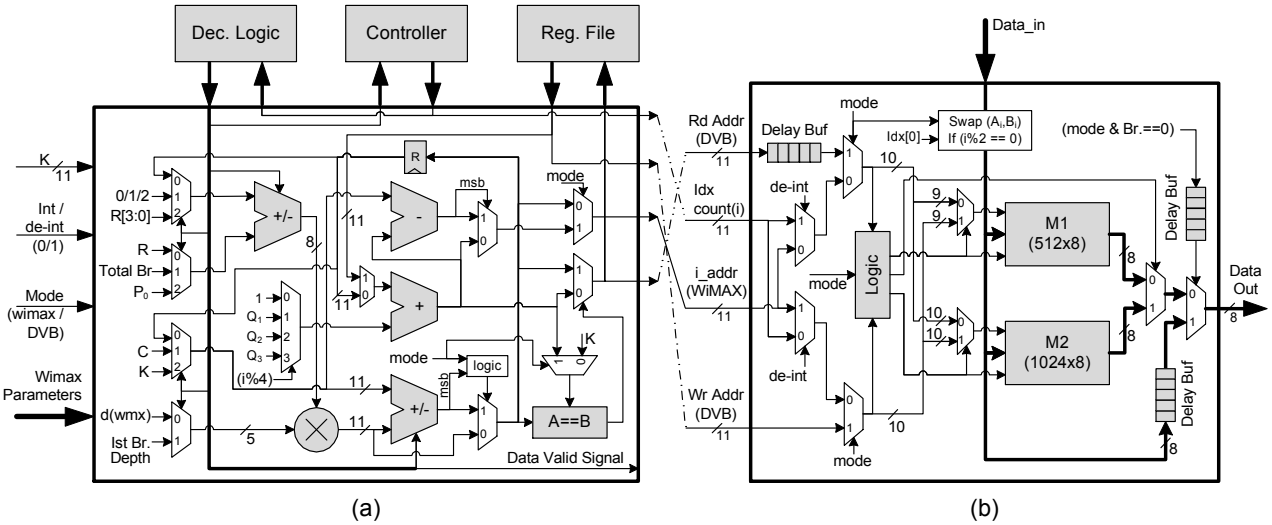

Fig. 11. (a) Address generation hardware for combined interleaver, (b) Memory organization for the hardware interleaver.

C. Memory Organization

The interleaved address for block interleaver and read/write address for convolutional interleaver computed by the address generation circuitry are combined according to the configuration input for specific standard to make the final read/write address for the memory. The total data memory size required is 1536 byte. This limit is set by maximum block size of 1536 bytes for WiMAX, i.e. 64-QAM and 16 sub-channels. If we have the luxury to use dual port memory which can read and write in a single clock cycle, then we can use one big memory of 1536 bytes, but keeping in view hardware in-efficiency for the implementation of dual port memory, and to make the memory size raise to the power 2, we split the memory into two memories with size 512 bytes and 1024 bytes.

Fig. 11(b) shows the memory organization with address selection logic. By applying the delay line of 5 clock cycles in the path of read address and control signal for the selection of the output data, we make it possible that data write and read should not be performed for the same memory in a single clock cycle. This provided the basis to use relatively bigger memories for DVB interleaver and thus the hardware cost overheads associated with use of small memories are also avoided. The memory utilization for DVB in comparison with use of small memories is shown in Fig. 12. The memory with size of 512 bytes as shown in Fig. 12(c) can be reduced to 256 bytes if proposed architecture is only intended to be used for DVB or the target block size for WiMAX is upto 1280 bytes.

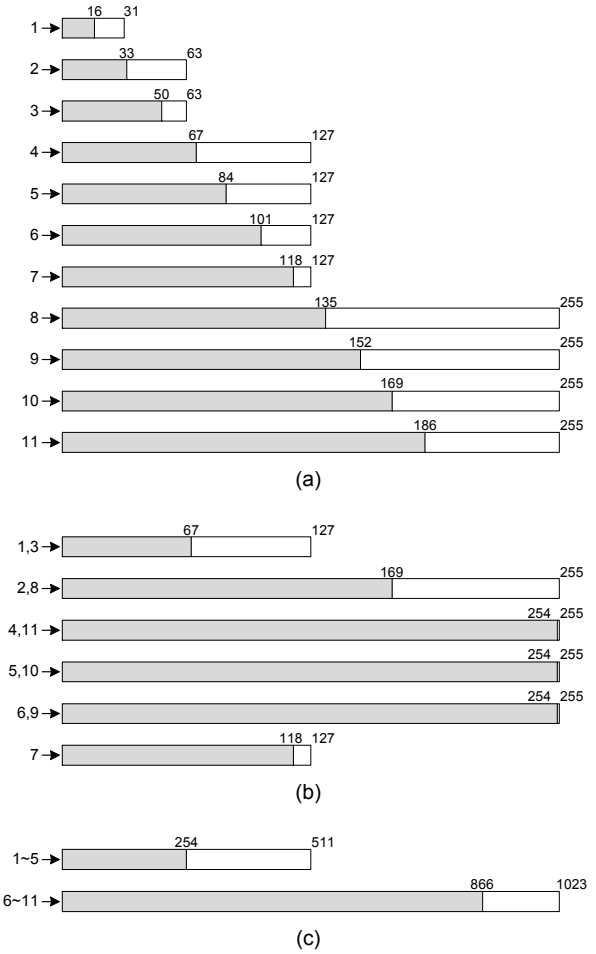

Fig. 12. Memory utilization for DVB (a) Generalized structure, (b) Structure proposed in [7], (c) Our proposed structure.

VI. Implementation Results

The hardware shown in Fig. 11 provides the complete re-configurable hardware interleaver design for multiple standards. The RTL code for this hardware is written in Verilog HDL and the correctness of the design is verified by two approaches. First by comparing the data from hardware with that of the interleaved data generated through MATLAB and then by checking the data in order, by cascading two hardware blocks, first configured as interleaver and second configured as de-interleaver.

TABLE I
HW USAGE COMPARISON FOR INTERLEAVER IMPLEMENTATION

No.	Implementation	Data Memory Structure	Total Memory Size	Total Size	Operating Frequency
1.	RAM for WiMAX (big off-chip Mem. required)	16.9K bit for addressing table + 12.2K bit for Data	28.5 Kbit	---	---
2.	RAM for DVB (General Structure) Extra Sync. Circuitry Needed	32 x 8b x 1 64 x 8b x 2 128 x 8b x 4 256 x 8b x 4	13.25 Kbit	---	---
3.	Xilinx Virtex-5 [18]	Block RAM 2048 x 9b x 1	18 Kbit	210 LUTs + memory	262/360 MHz Speed Grade -1/-3
4.	Altera FLEX-10KE [19]	8 Embedded Array Blocks each 2048 bits	16 Kbits	392 LEs + memory	120 MHz
5.	Lattice ispXPGA [20]	8 Block RAMs 512 x 9b x 8	36 Kbits	284 LUTs + memory	132 MHz
6.	Ref. Design [4]	256 x 8b x 6	12 Kbit	0.60 mm^2	100 MHz
7.	Ref. Design [5]	256 x 8b x 6	12 Kbit	0.484 mm^2	150 MHz
8.	Ref. Design [6]	12 x 24b x 6 108 x 36b x 8	32 Kbit	0.72 mm^2	200 MHz
9.	Ref. Design [7]	128 x 8b x 2 256 x 8b x 4	10 Kbit	---	---
10.	Our Design	512 x 8b x 1 1024 x 8b x 1	12 Kbit	0.18 mm^2	140 MHz

After validating the correctness of design, the design was synthesized for 0.12µm standard CMOS technology and then layout was generated using SoC Encounter. Core size of the proposed architecture is $0.18\,mm^2$ which is lower than the reference designs as shown in Table 1. The major gain in terms of silicon cost is due to efficient and shared implementation of address computation circuitry and by sharing the data memory as shown in Fig. 13. Although direct comparison of area with commercially available FPGA implementations [18] – [20] is not possible, but looking at the memory requirements our design remains efficient. The available literature reveals that these commercial implementations use dual port memory support from block RAMs available on target FPGA platforms, which is not a good choice for chip implementations. Further, it also reveals that they do not compute the row or column permutations on the fly; instead they take row or column permutation tables in the form of a configuration file as input and use them to generate the final interleaved address. In this way, the complexity for on-the-fly computation of permutation patterns is avoided but it requires extra memory to store the permutation patterns.

Chip core layout for the proposed architecture is shown in Fig. 14. The design can run upto 140 MHz without any pipelining and consumes 3.5mW power in total. Introducing pipeline stages can further enhance the performance to make it a good choice for future high speed communication systems. The interleaved data from block interleaver for WiMAX/WLAN is provided in every clock cycle. However, if the block size is selected in such a way that it is not exactly equal to the size of row-column matrix, then zero padding is used for the un-used spaces and these zeros are pruned out while reading the data from matrix after all permutations. If pruning is needed then maximum of two clock cycles are needed to get a valid data output from interleaver/de-interleaver.

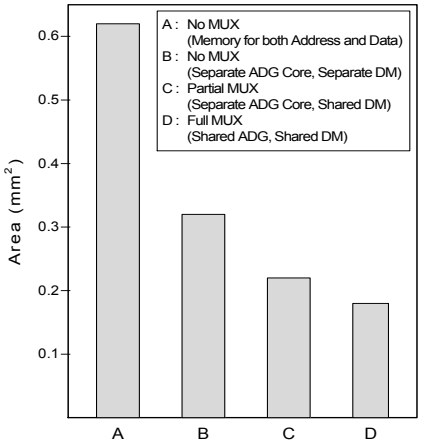

Fig. 13. Cost comparison for hardware multiplexing.

VII. CONCLUSION

A very low silicon cost hardware interleaver for WiMAX/WLAN and DVB is presented which supports the mapping of vital types of interleavers like channel interleaver, CTC block interleaver and convolutional interleaver onto a single architecture. The realization of low cost solution for interleaver address generation is made possible by reducing the algorithmic complexity by re-

structuring the interleaving algorithms to avoid complex functions. Transformation of 1-D interleaver functions into 2-D space provided the major breakthrough to reduce the hardware cost. At the same time, the hardware overheads due to small memory banks are minimized by using relatively larger memories. The final results presented in Table 1, show the silicon efficiency in comparison with reference designs. Reduced complexity encourages enabling on-the-fly computation of permutation patterns for interleaver.

The presented architecture consumes 3.5mW power at a frequency of 140 MHz, thus providing sufficient performance for high speed communication. It also provides a generalized platform to map different block interleavers with different block sizes and also to map different convolutional interleavers having upto 12 branches and requiring a total memory less than or equal to 1536 bytes. Performance, low silicon cost and re-configurability for multiple standards make this architecture a suitable candidate to be adapted in many multimode communication systems.

ACKNOWLEDGMENT

The authors would like to thank Anders Nilsson, Eric Tell, and Erik Alfredsson of Coresonic AB, Linköping, Sweden, Johan Eilert and Di Wu of LiU, Linköping, Sweden, for some helpful discussions on interleaver design and tool setup.

REFERENCES

[1] IEEE 802.16e–2005: "IEEE Standard for local and metropolitan area networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems – Amendment 2: Medium Access Control Layers for Combined Fixed and Mobile Operations in Licensed Bands."

[2] ETSI EN 300-744 V1.5.1: "Digital Video Broadcasting (DVB); Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television," Nov. 2004.

[3] IEEE 802.11-2007: "Standard for local and metropolitan area networks, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Sepcifications," Revision of IEEE Std. 802.11-1999.

[4] Y. N. Chang and Y. C. Ding: "A Low-Cost Dual Mode De-interleaver Design," Int. conf. on Consumer Electronics, 2007.

[5] Y. N. Chang: "A Low-Cost Dual Mode De-interleaver Design," IEEE Transaction on Consumer Electronics, vol. 54, no. 2, May 2008, pp. 326 – 332.

[6] Y. W. Wu and P. Ting: "A High Speed Interleaver for Emerging Wireless Communications," Proc. of International Conf. on Wireless Networks, Communications and Mobile Computing, vol. 2, June 2005, pp. 1192 – 1197.

[7] J. B. Kim, Y. J. Lim and M. H. Lee: "A low complexity FEC Design for DAB," Proc. of IEEE Int. Symposium On Circuits and Systems, May 2001, vol. 4, Sydney, Australia, pp. 522 – 525.

[8] G. D. Forney: "Burst-Correcting Codes for the Classic Bursty Channel," IEEE Transaction on Communication Tech., Oct-1971, Vol. COM-19, No. 5, pp. 772 – 781.

[9] J. L. Ramsey: "Realization of Optimum Interleavers," IEEE Transaction on Information Theory, May-1970, Vol. IT-16, No. 3, pp. 338 – 345.

[10] C. Berrou, A. Glavieus, and P. Thitimajshima: "Near Shannon limit error-correcting coding and decoding: Turbo-codes," Proc. of IEEE ICC, May 1993, vol. 2, pp. 1064 - 1070.

[11] Ji-Hoon Kim and In-Cheol Park: "Duo-binary circular turbo decoder based on border metric encoding for WiMAX," Proc. of IEEE ASPDAC, March 2008, pp. 109 – 110.

[12] Cheng-Hung Lin, Chun-Yu Chen and An-Yeu Wu: "High-Throughput 12-Mode CTC Decoder for WiMAX Standard," Proc. of IEEE VLSI-DAT, April 2008, pp. 216 – 219.

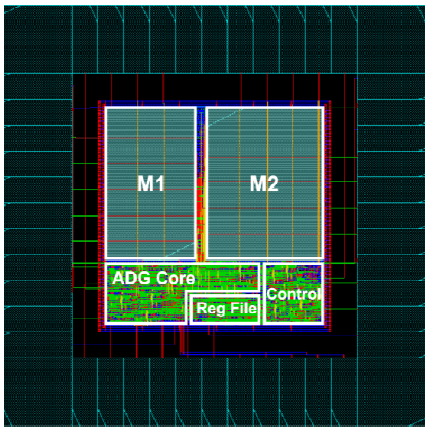

Fig. 14. Layout snapshot of proposed interleaver architecture.

[13] C. Berrou, M. Jezequel, C. Douillard, and S. Kerouedan: "The advantages of non-binary Turbo codes," Proc. of IEEE Info. Theory Workshop, Sept. 2001, pp. 61–63.

[14] S. Vafi and T. Wysocki: "Weight distribution of turbo codes with convolution interleavers," IET Communications, 2007, Vol. 1(1), pp. 71 – 78.

[15] E. K. Hall and S. G. Wilson: "Stream-Oriented Turbo Codes," IEEE Transaction on Information Theory, July-2001, Vol. 47, No. 5, pp. 1813 – 1831.

[16] S. Vafi and T. Wysocki: "On the Performance of Turbo Codes with Convolutional Interleavers," Proc. of Asia-Pacific Conference on Communications, Oct. 2005, pp. 222 – 226.

[17] S. Vafi and T. Wysocki: "Performance of convolutional interleavers with different spacing parameters in turbo codes," Proc. of 6th Australian Communication Theory Workshop, Feb. 2005, pp. 8 –12.

[18] Xilinx Inc.: "Interleaver/De-Interleaver," Product Specification, v5.1, DS250, March 2008.

[19] Altera Inc.: "Symbol Interleaver/De-Interleaver Core," Mega Core Function User's Guide, ver. 1.3.0, June 2002.

[20] Lattice Semiconductor Inc.: "Interleaver/De-Interleaver IP Core," Core User's Guide, ipug_61_02.5, August, 2008

R. Asghar is working on a Ph.D. degree at the Department of Electrical Engineering of Linköping University, Sweden. He received M.Sc. degree in Physics from Quaid-i-Azam University, Islamabad, Pakistan, and M.S. degree in Computer Engineering from Center for Advanced Studies in Engineering (CASE), Islamabad, affiliated with University of Engineering and Technology, Texila, Pakistan. His research activity is mainly focused on flexible and re-configurable forward error correction sub-systems for baseband processor platform.

D. Liu is Professor and the Director of Computer Engineering at the Department of Electrical Engineering of Linköping University, Sweden. He got technology doctor degree from Linköping University Sweden in 2005. He is IEEE senior member. He published more than 100 papers and holds 5 US patents. The focus of his research is high performance low power ASIP (application specific instruction set processors) and integration of on-chip multi-processors for communications and media signal processing. Dake has experiences also in design of communication systems, Radio frequency CMOS integrated circuits. Dake Liu is the co-founder and CTO of FreehandDSP AB Stockholm Sweden. Dake Liu is currently the co-founder and CTO of Coresonic AB Linköping Sweden. Dake Liu was a senior ASIC designer and low power design specialist in Ericsson Microelectronics, Stockholm since 1995 to 1998.