Memory Conflict Analysis and Implementation of a Re-configurable Interleaver Architecture Supporting Unified Parallel Turbo Decoding

Rizwan Asghar, Di Wu, Johan Eilert and Dake Liu

Memory Conflict Analysis and Implementation of a Re-configurable Interleaver Architecture Supporting Unified Parallel Turbo Decoding

Rizwan Asghar, Di Wu, Johan Eilert, Dake Liu

Abstract − This paper presents a novel hardware interleaver architecture for unified parallel turbo decoding. The architecture is fully re-configurable among multiple standards like HSPA Evolution, DVB-SH, 3GPP-LTE and WiMAX. Turbo codes being widely used for error correction in today's consumer electronics are prone to introduce higher latency due to bigger block sizes and multiple iterations. Many parallel turbo decoding architectures have recently been proposed to enhance the channel throughput but the interleaving algorithms used in different standards do not freely allow using them due to higher percentage of memory conflicts. The architecture presented in this paper provides a re-configurable platform for implementing the parallel interleavers for different standards by managing the conflicts involved in each. The memory conflicts are managed by applying different approaches like stream misalignment, memory division and use of small FIFO buffer. The proposed flexible architecture is low cost and consumes 0.085 mm^2 area in 65nm CMOS process. It can implement up to 8 parallel interleavers and can operate at a frequency of 200 MHz, thus providing significant support to higher throughput systems based on parallel SISO processors.

Keywords − Parallel interleaver, Parallel turbo decoding, Block interleaver, Multi standard, HSPA, LTE, WiMAX, DVB-SH.

1 Introduction

Latest trends in radio communication systems always demand for a flexible and general purpose solution for data processing including symbol processing and forward error correction (FEC). Turbo Codes [1] are being widely used as a forward error correction system in the consumer electronics in order to detect and correct the errors during transmission over noisy channels. Interleaving plays a vital role in improving the performance of Turbo Codes in terms of bit error rate. The primary function of the interleaver is to improve the distance properties of the coding schemes and to disperse the sequence of bits in a bit stream so as to minimize the effect of burst errors. At present a number of standards specifications like HSPA Evolution [2][3], DVB-SH [4], UMTS-LTE [5] and WiMAX [6] have already included the turbo codes. The scheme of turbo code adapted is the parallel concatenated code with two 8-state constituent encoders and an internal interleaver. One turbo code internal interleaver is used in the turbo encoder while the turbo decoder uses multiple instances of interleaver and de-interleaver to decode the received bits iteratively (Figure 1).

Along with multi-standard support, requirements of higher throughput are also emerging in connection with customer needs. On the other hand turbo codes in general exhibit latency (further reducing throughput) due to bigger block sizes and multiple iterations over soft bits needed to reach to a reliable hard decision. The latency can increase further if interleaver block size is varying in every transmission time interval (TTI) with the requirement of some pre-processing while changing the block size. The technique of parallel turbo decoding is well established to meet the high throughput requirements [7] - [10], but it also requires implementation of parallel interleavers. If there are P parallel SISO processors then P sub-interleavers has to be implemented in parallel and each sub-interleaver will be responsible to generate N/P interleaver patterns independently where N is the total block size. The main schemes of parallel turbo decoding can be applied to different standards without any modifications; however, interleaving varies among different standards. Many challenges are involved including unified address generation and conflict management which restrict the use of same parallel turbo decoding architecture for multiple standards. This paper paves special focus on handling these challenges and presents an

Rizwan Asghar
Department of Electrical Engineering,
Linköping University,
Linköping, SE-58183, Sweden
Phone : +46 13 281323
FAX: +46 13 139282
e-mail: rizwan@isy.liu.se

Di Wu, Johan Eilert, Dake Liu
e-mail: diwu@isy.liu.se; je@isy.liu.se; dake@isy.liu.se

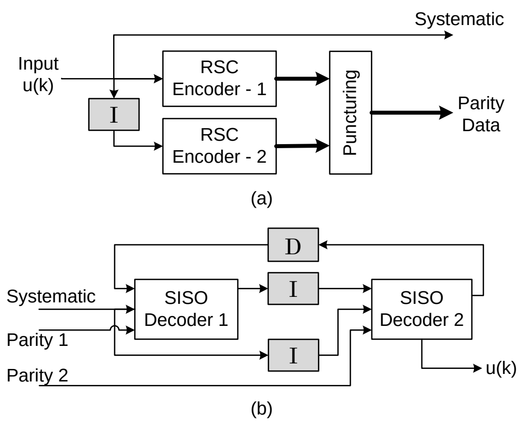

Figure 1: (a) Turbo Encoder; (b) Turbo Decoder.

implementation of a re-configurable interleaver architecture targeting unified parallel turbo decoding.

The rest of the sections in this paper are organized as follows. Section 2 provides a picture of previous work done and challenges involved while implementing parallel interleaver support for parallel turbo decoding. Sections 3 – 6 cover the parallel interleaving in HSPA Evolution, DVB-SH, 3GPP-LTE and WiMAX standards respectively. Following the hardware multiplexing methodology, the unified architecture of the re-configurable parallel interleaver is presented in Section 7. Section 8 provides the implementation results followed by a conclusion.

2 Background and Challenges

Looking at the implementation aspects recursive systematic convolutional encoders are very simple to implement as compared to SISO decoding in turbo decoder, but the interleavers usually tend to have same complexity on both sides because of its variability. The implementation of interleaver on decoder side becomes more challenging when parallel interleaver implementation is needed in order to support parallel SISO processing. Other then the architectural complexity for generation of parallel interleaver addresses one of the big challenges is to deal with memory conflicts associated with multiple writes and reads at the same time. If the multiple addresses generated do map to different memories i.e. one address for each memory then there is no conflict, but on the other hand if two or more addresses are mapped to same memory then a situation of conflict occurs (Figure 2) and it needs to be resolved on-the-fly.

The conflict percentage from the parallel generated addresses is sometimes very high and no straight forward solution exists except using double memory size which is not cost efficient. Work in [11] − [13] provide good theoretical back ground and also propose the generation of conflict free

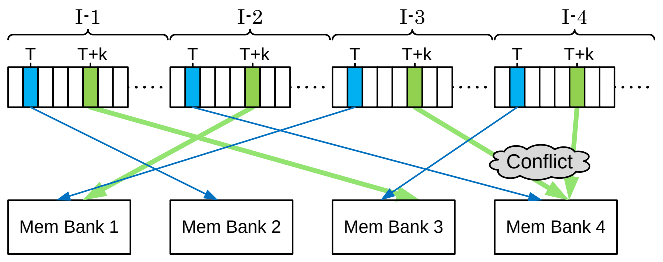

Figure 2: A situation of conflict at (T+k).

interleavers but they cannot be directly used for already existing interleaver algorithm for the standards being investigated in this paper. The work presented in [14] − [19] covers single address generation supporting maximum of two standards but they do not support parallel interleaver address generation. Reference [21] provides a parallel interleaver architecture for HSPA+ but it does not cover the support for multi-mode environment. A good analysis of memory conflicts for interleaver in turbo decoder is provided in [22]. The stalling mechanism is used to stall the process in MAP decoders when a conflict occurs. The main focus of this work has been the buffer management. Further the stall process may result in additional control complexity. Reference [23] provides a good theoretical background along with an interleaver architecture supporting parallel turbo decoding. However, it supports only one standard i.e. WiMAX Duo-Binary Turbo Decoding. The implementation of a parallel interleaver for a multi-mode environment appears to be a bottleneck, which also restricts to use same turbo decoding core for multiple standards. This motivates the work on a re-configurable interleaver architecture to support unified parallel turbo decoding. A low cost and re-configurable solution for parallel interleaver implementation for existing interleaver algorithms can open more opportunities to meet the high throughput requirements and at the same time it can help to meet fast time-to-market requirements from industry and customers. This paper addresses the management of memory conflicts and proposes certain schemes to reduce them with lower silicon cost overheads. By exploiting the hardware re-use methodology among different standards a unified architecture is presented which is low cost and fully re-configurable to generate the parallel interleaving patterns on-the-fly.

3 Parallel Interleaver for HSPA Evolution

The throughput requirements for WCDMA based systems have been raised in a series of specifications. After addition of MIMO and 64 QAM in Release 8 [3] the throughput requirements have reached to 43.2 Mbps. The interleaver associated with HSPA+ is not inherently designed to support parallel SISO processing, but higher throughput requirements and requirement to change the block size in each transmission time interval motivates to have the provision of more than one SISO processing in parallel. The interleaving algorithm for 3GPP-WCDMA is mentioned below. Here N is the block size, R is the row size and C is the column size in bits.

- *Find number of rows 'R', prime number 'p' and primitive root 'v' for particular block size as given in the standard.*
- *Col Size :* $C = p-1;$ *if* $(N \leq R \times (p-1))$
  $C = p;$ *if* $(R \times (p-1) < N \leq R \times p))$
  $C = p+1;$ *if* $(R \times p < N)$
- *Construct intra row permutation sequence S(j) by:*
  $S(j) = [v \times S(j-1)] \% p ; \quad j = 1,2, \ldots\ldots p-2$
- *Determine the least prime integer sequence q(i) for i = 1,2, …… R-1 , by taking q(0) = 1, such that g.c.d(q(i),p-1) = 1 and q(i) > 6 and q(i) > q(i-1).*
- *Apply inter row permutations to q(i) to find r(i) = T (q(i))*
- *Perform the intra row permutations Ui,j; for i = 0,1,... R-1 and j = 0,1, ... p-2;*
  If (C=p) : $U_{i,j} = S [(j \times r(i)) \bmod (p-1)]$ *and* $U_{i,(p-1)}=0;$
  If (C=p+1) : $U_{i,j} = S [(j \times r(i)) \bmod (p-1)]$ *, and* $U_{i,(p-1)} = 0; U_{i,p} = p;$ *and if* $(N = R \times C)$ *then exchange U(R-1,0) with U(R-1,p)*
  If (C=p-1) : $U_{i,j} = S [(j \times r(i)) \bmod (p-1)] - 1 ;$
- *Perform the inter row permutations*
- *Read the address columns wise*

The complication in the implementation is evident due to the presence of complex functions like modulo computation, intra-row and inter-row permutations, multiplications, finding least prime integers, and computing greatest common divisor. After applying the intra-row and inter-row permutations a block of random addresses denoted by y_k appears as shown below:

$$\begin{bmatrix} y_1 & y_{(R+1)} & y_{(2R+1)} & \cdots & y_{((C-1)R+1)} \\ y_2 & y_{(R+2)} & y_{(2R+2)} & \cdots & y_{((C-1)R+2)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ y_R & y_{2R} & y_{3R} & \cdots & y_{(C\times R)} \end{bmatrix}$$

The output from the interleaver is the sequence read column-wise from the permuted matrix. The output address Y_k within the matrix can be expressed as:

```
loop $i : 1$ to $C$
  loop $j : 1$ to $R$
        $Y_k = y_{(i-1)R+j}$
```

Generating two addresses Y_k^1 and Y_k^2 at the same time will reduce the overall loop size to half as follows:

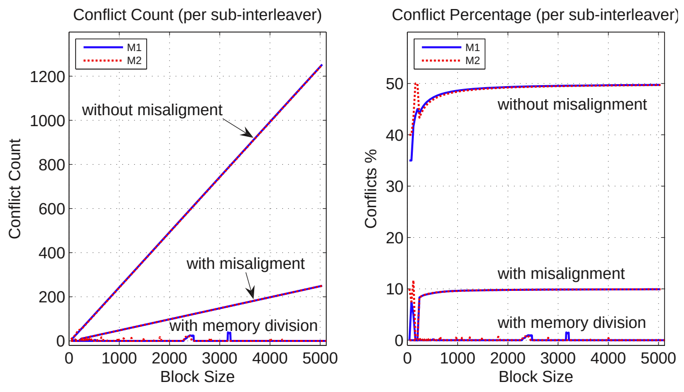

Figure 3: Conflict analysis for HSPA+.

```
loop $i : 1$ to $\frac{C}{2}$
  loop $j : 1$ to $R$
        $Y_k^1 = y_{(i-1)R+j}$
        $Y_k^2 = y_{\left(\frac{2(i-1)R+RC}{2}+j\right)}$
```

The two addresses generated simultaneously are used to write two data values in two memory locations. In case of a conflict while writing into same memory, it needs to be resolved on-the-fly. The following sub-section provides a comprehensive analysis to reduce the memory conflicts associated with whole range of block sizes in HSPA+.

3.1 Memory Conflict Analysis

The number of conflicts for HSPA+ interleaver reaches to 50% of the data size as shown in Figure 3. One way to reduce the number of conflicts is to introduce a misalignment in the generation of two addresses (Figure 4). The misalignment can be achieved by introducing a delay line to one set of addresses and data values. It introduces the latency on one of the two sequences, thus the arbitrary value called misalignment factor (MF) cannot be very large. There are three possible values for R (number of rows) in the prescribed interleaver i.e. 5, 10 or 20 and good MF must be within the total number of rows. It is observed that the inter-row permutation patterns for R=5 or 10 are supportive to misalignment technique but the permutation patterns for R=20 are not very much supportive. The best MF found is 3 for R=5 or R=10 and 5 for R=20. Using these values of MF the conflicts are reduced to around 10 % as shown in Figure 3. Still the conflict count is high enough that it cannot be managed without the support of extra memories. Alternately, a large amount of buffer registers can be used to handle this situation but it involves higher latency.

Another approach might be to split the memory banks into relatively smaller sub-memories to reduce the number of

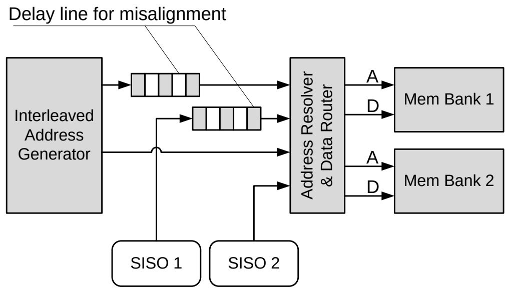
Delay line for misalignment
Interleaved Address Generator
Address Resolver & Data Router
A
D
Mem Bank 1
A
D
Mem Bank 2
SISO 1
SISO 2

Figure 4: Misalignment of address and data to reduce memory conflicts.

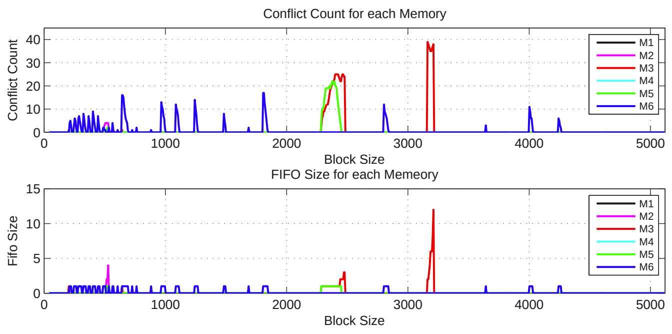

Figure 5: Final conflict count and FIFO size requirement for HSPA+ interleaver.

conflicts. Up to 8 memories are already needed for 8 parallel turbo interleavers in this design, thus we can divide each memory bank required for HSPA+ into 3 sub-memories. By applying this configuration to the interleaver address generation and data writing the amount of conflicts for most of the block sizes reduced to zero (Figure 3), but still many block sizes face some conflicts. However, the amount of conflicts is very small and it can be handled by using buffer registers. The number of FIFO buffers can further be reduced by applying the progressive write during the conflict occurrences in the other memory bank. The total number of FIFO buffers needed for each memory after memory division is plotted in Figure 5 with maximum of 12 FIFO registers required for memory M3.

3.2 Pre-Processing

In order to make the interleaver architecture fully autonomous, the parameters like R, C , $q(i)$, $T(i)$, $S(j)$, p and v are needed to be computed in hardware. Some of these parameters can be computed using lookup tables while the others need some close loop or recursive computations. The most critical parameter consuming more clock cycles and more hardware to compute is the intra-row permutation pattern $S(j)$. The function to be computed for all the permutations is:

$$S(j) = \{v \times S(j-1)\} \% p \tag{1}$$

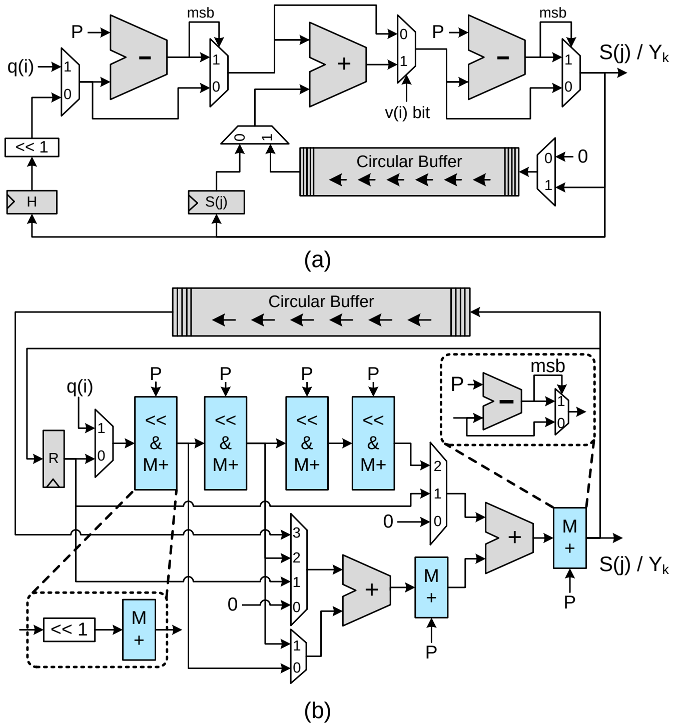

Figure 6: Hardware for computing S(j) using (a) Interleaved modulo multiplication algorithm, (b) Segmentation based modulo computation.

Table 1: HSPA+ Pre-processing cycle cost comparison (cycles).

Block Size	Ref. [14]	Ref. [16]	Ref. [18]†	Proposed
40	317	15	19	11
41	295	23	31	14
5040	3587	802	821	303
5114	3048	563	583	310

† Values computed by using the methodology given in [18]

Where $j = 1, 2, ...p-2$. This function exhibits un-known clock cycle delay due to the reason that the value $S(j-1)$ is not known with each j, therefore targeting low cost implementation it is computed beforehand recursively and the results are stored in a small RAM called intra-row permutation RAM (IRP_RAM). One way of computing this parameter is using the Interleaved Modulo Multiplication Algorithm [20] which needs more than one clock cycle to compute one value. The hardware to compute $S(j)$ using modulo multiplication algorithm is shown in Figure 6 (a). Looking at the maximum size of v which is 5 bits this algorithm can take 5 clock cycles to compute each $S(j)$ recursively.

Another approach to compute the modulo function is the segmentation based modulo computation (SBMC). The idea is to use only addition functions or shift-left by one in series followed by a modulo addition. This method might not be

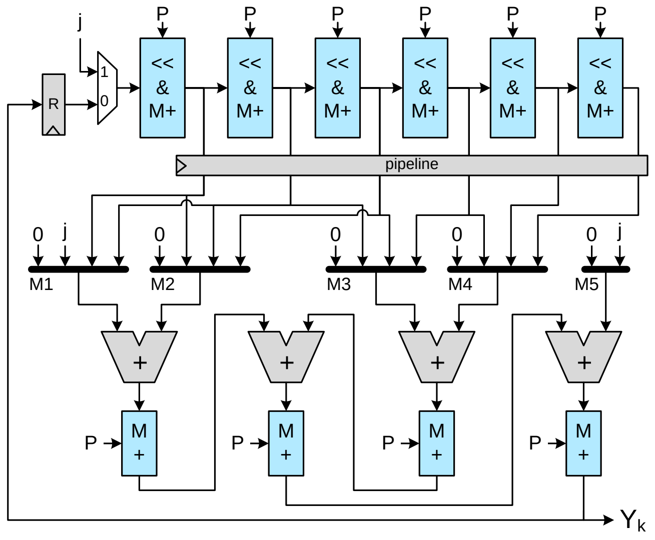

Figure 7: Computing RAM address (*RA*) using SBMC approach.

hardware efficient when the multiplication terms are wide range; however, it is beneficial if any one of the multiplication term is relatively smaller in range. Here parameter v is having limited number of values i.e. 2,3,5,6,7 or 19. Thus SBMC can be optimized to achieve low cost solution. The hardware for computing the intra-row permutation using SBMC is shown in Figure 6(b). The main advantage of using this scheme over Modulo Multiplication Algorithm is the single clock cycle execution for finding each new $S(j)$, which gives a big saving of pre-processing cycle cost over the previous proposed designs (see Table 1). While supporting the generation of two interleaved addresses per clock cycle, two parallel IRP_RAMs each having size 256 x 8b are required to store the intra-row permutation patterns. Keeping in view the hardware in-efficiency only one can be used with size 128x16b and the 16 bit data output is further divided to be used for two different sections of address generation. The second address with which the data has to be merged is computed by:

$$j_{aux} = \left(j + \frac{C}{2} \right) \%\ p \qquad (2)$$

The parameters j and j_{aux} are compared and smaller one is used to write the data. The same expression and comparison is used to resolve the data distribution after reading the data. The other parameters to be computed in the pre-processing phase are prime number p, associated integer v and total number of columns C. The parameter p is stored in a lookup table with a smaller sub-lookup table for parameter v. The lookup table is addressed via a counter and against each value of p the condition ($p \times R \geq N - R$) is tested using a comparator. Once p is found, C can have only three values i.e. $p-1$, p or $p+1$, thus

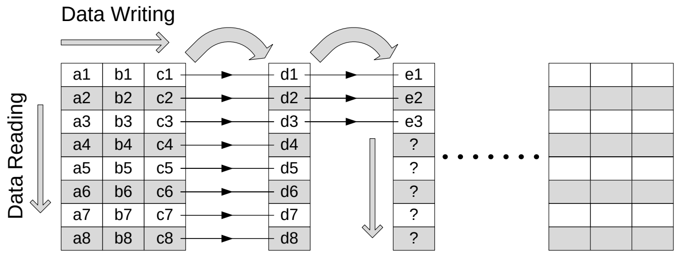

Figure 8: Column by column recursive address computation.

requiring at most three clock cycles. The hardware used during the pre-processing phase remains idle during the execution phase; therefore it can be reused to reach to a low cost solution.

3.3 HW for Parallel Interleaver in HSPA+

The parallel processing of turbo decoder with parallel SISO blocks requires the interleaver to generate more than one address every clock cycle and at the same time resolve the memory conflicts. The computation of final interleaved address requires that the intra-row permutation data must be obtained in correct order from IRP_RAM. The data output from the RAM is denoted as $U(i,j)$ and it is given by:

$$U(i,j) = S\left[j \times r(i) \% (p-1) \right] \qquad (3)$$

The address (*RA*) to the RAM i.e. $j \times r(i) \% (p-1)$ involves modulo function. We present here three alternates to compute the RAM address. The first method involves recursive computation of addresses as shown in Figure 8. The data is written into memory row wise but when reading back column wise, the IRP_RAM address of previous column is used to find next address. The recursive function is given by:

$$RA(i,j) = \left\{ RA(i,j-1) + qmod(i) \right\} \% (p-1) \qquad (4)$$

The parameter $qmod(i)$ is computed from the least prime numbers sequence $q(i)$ by $qmod(i) = q(i)\%(p-1)$. With the condition i.e. $q(i) < 2(p-1)$ only a subtraction can be used to obtain $qmod(i)$. The computational part associated with recursive approach is very small and limited to just few adders, but it needs a circular buffer of size 20 x 8b (Figure 6 (a)) in order to keep the old address of whole column. This approach is low cost and can operate at very high clock rate due to smaller critical path. Using the same hardware for computing $S(j)$ requires five clock cycles. To reduce it to single clock cycle we need to use the mix of the recursive approach and the segmentation based modulo computation (Figure 6(b)). This second approach is not very much hardware efficient but gives the benefit to reduce the pre-processing time.

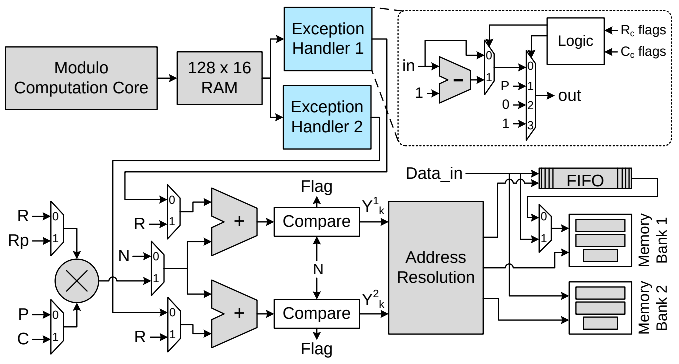

Figure 9: HW for parallel interleaver address generation in HSPA+.

The third approach is completely based on segmentation based modulo computation and it is a non-recursive way to compute the address RA. It can directly be applied to the term $j \times r(i)\%(p-1)$ to get the address for register file. Here we know that the parameter $r(i)$ can have only 22 values of prime numbers up to 89, thus SBMC approach can be used after applying some optimizations. The hardware required to compute the function $j \times r(i)\%(p-1)$ using SBMC approach is shown in Figure 7. It needs more additions then the recursive approach; however, it can directly be used to compute intra-row permutation patterns in the pre-processing phase, thus providing single clock cycle support for computing each permutation pattern $S(j)$. This approach cannot be better for very high clock frequency due to longer critical path. Pipelining can improve the performance of this scheme but at the same time introduces higher control complexity.

The computed address is used to get the correct intra-row permutation $U(i,j)$ from the RAM. It also needs to pass through some exception handling logic to correct itself for special cases associated with $C = p$, $C = p+1$ or $C = p-1$ as given in the algorithm. The final interleaved addresses $Y^1_{i,j}$ and $Y^2_{i,j}$ can be found by combining the inter-row permutation with intra-row permutation as follows:

$$Y^1_{i,j} = \{C \times r(i)\} + \mathrm{U}^1(i,j) \tag{5}$$

$$Y^2_{i,j} = \{C \times r(i)\} + \mathrm{U}^2(i,j) \tag{6}$$

The complete hardware for the generation of parallel interleaved addresses and data handling is shown in Figure 9. A FIFO buffer of size 12 is needed as discussed in previous sub-section to handle the conflicts. The generation of interleaved address during run time is two addresses per clock cycle, except the case when block size is not exactly equal to $R \times C$. In this case data written into RAM is zero padded and data pruning is performed using the comparators.

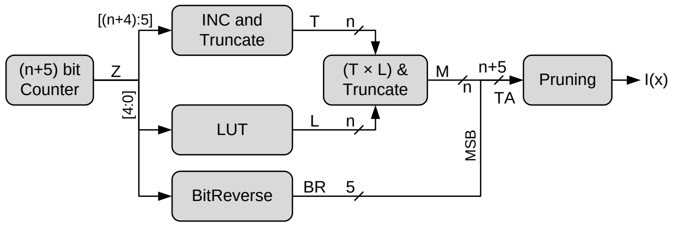

Figure 10: Interleaver address computation flow for DVB-SH.

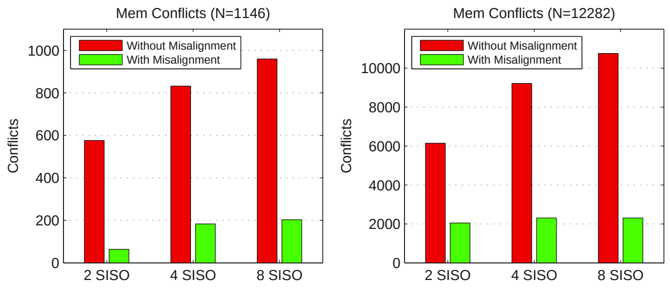

Figure 11: Total conflicts with and without misalignment (DVB-SH).

4 Parallel Interleaver for DVB-SH

With improvements in DVB-H, DVB-SH standard specification provides satellite services to handheld devices. Along with other improvements, the FEC is also improved with inclusion of turbo coding. The interleaving associated with turbo code is to write whole block of information sequentially into an array and reading the information in an order defined by the interleaving algorithm. The flow for interleaver address computation is shown in Figure 10. There are only two possible block sizes i.e. N = 1146 or N = 12282 with parameter n having condition as the largest value such that $N \le 2^{n+5}$. Two lookup tables each having 32 entries, are also provided to be used in interleaving. The interleaving algorithm can be described as follows:

- *Find the parameter n such that* $N \le 2^{n+5}$.
- *Initialize an (n+5) bit counter (called Z here).*
- *Find T = Truncate (Z[(n+4):5]+1) to n bits.*
- *Obtain the table lookup output (L); L = LUT{Z[4:0]}.*
- *Multiply and truncate to find M = Truncate (T × L) to n bits.*
- *Find bit reverse BR = BitReverse{Z[4:0]}.*
- *Find Tentative Address by concatenation TA = {BR, M}.*
- *Compare the TA and prune out the addresses if* $TA \ge N$

The latency in turbo coding can be higher due to bigger block size of 12282 which demands the provision of parallel SISO

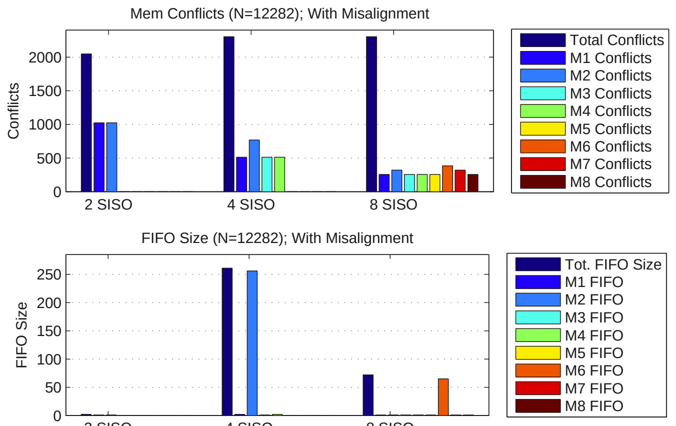

Figure 12: Memory conflicts and FIFO size requirement for DVB-SH (N=12282; with misalignment).

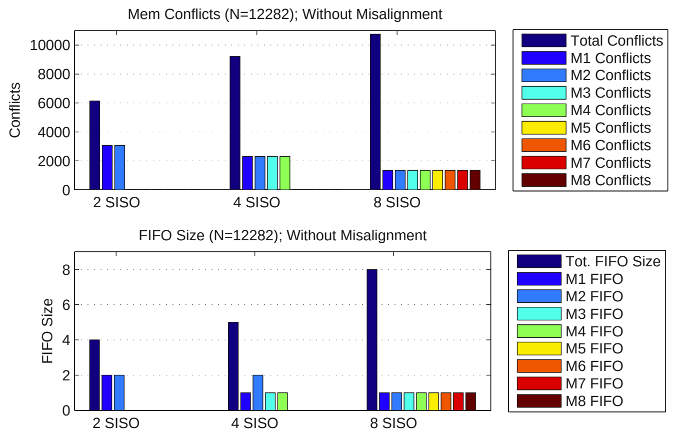

Figure 13: Memory conflicts and FIFO size requirement for DVB-SH (N=12282; without misalignment).

processing. The parallel address generation involves a number of conflicts which need to be resolved on-the-fly. The final architecture proposed can support up to 8 parallel SISO blocks. The following sub-sections provide the detailed conflict analysis and the hardware for parallel interleaver address generation in DVB-SH.

4.1 Memory Conflict Analysis for DVB-SH Interleaver

The parallel generated interleaved addresses for DVB-SH are not conflict free. Figure 11 shows the number of conflicts for the two blocks sizes. Applying the method of misaligning on the address and data streams, it is observed that the conflict percentage reduces significantly.

Further analysis of conflicts (N=12282 only) on each memory is provided in detail in Figure 12 and Figure 13. In contrary to conflict reduction due to applying misalignment the number of FIFO registers required to handle the number of conflicts are more. The analysis reveals that the distribution of conflicts for a particular memory over a range of block size becomes irregular with misalignment which increases the requirement of FIFO cells. On the other hand although the number of conflicts are huge but they are uniformly distributed among different memories and hence it becomes more hardware efficient to use the scheme without misalignment.

Table 2: Lookup Table for Basic and Parallel Address Generation (DVB).

No	N = 1146				N = 12282			
(j)	Effective R_c	T_{bas}	T_{aux}	*Inc.*	Effective R_c	T_{bas}	T_{aux}	*Inc.*
0	0	3	3	1	0	13	5	1
1	1	27	3	1	1	335	7	1
2	2	15	7	2	2	87	7	2
3	4	29	5	2	4	15	7	1
4	6	1	1	2	5	1	1	1
5	8	3	3	2	6	333	5	2
6	10	15	7	2	8	13	5	1
7	12	17	1	2	9	1	1	1
8	14	39	7	2	10	121	1	2
9	16	19	3	1	12	1	1	1
10	17	27	3	1	13	175	7	1
11	18	15	7	2	14	421	5	2
12	20	45	5	2	16	509	5	1
13	22	33	1	2	17	215	7	1
14	24	13	5	2	18	47	7	2
15	26	15	7	2	20	295	7	1
16	28	17	1	2	21	229	5	1
17	30	15	7	2	22	427	6	2
18	--	--	--	--	24	409	1	1
19	--	--	--	--	25	387	6	1
20	--	--	--	--	26	193	1	2
21	--	--	--	--	28	501	5	1
22	--	--	--	--	29	313	1	1
23	--	--	--	--	30	489	1	2

4.2 HW for Parallel Interleaver in DVB-SH

As there are many invalid addresses computed by the algorithm so the total number of clock cycles needed to complete one block is much high than the block size. The simulation reveals that the block size N=1146 needs 2045 cycles whereas N=12828 needs 16382 cycles to get the valid values equal to block sizes. Our first target is to reduce the invalid computations and then generate the parallel addresses to further enhance the throughput. Instead of considering a long array of information we consider it as a block of information with total number of rows R as 32 (corresponding to 32 entries in the lookup table) and total number of columns C as 64 and 512 for N = 1146 and 12282 respectively. The generation of valid addresses within one row appears to be very much systematic which can be utilized to reduce the number of invalid addresses. Simulation results have proven that about 14 computations for N = 1146 and 8 computations for N = 12282 within a row can be completely discarded. This also reduces the lookup table requirement to 18 and 24 entries for N = 1146

Algorithm 1: Modified Algorithm for Parallel Address Generation in DVB-SH.

Initialization:

```
if      $N = 1146$       : $n = 6; R_T = 18; C_T = 64;$
elseif  $N = 12282$      : $n = 9; R_T = 24; C_T = 512;$
$R_c(0) = 0;$
loop $p$ : 1 to $R_T$
        $M_1(0, p) = 0$
end : loop $p$
```

Execution:

```
loop $i$ : 1 to $C_T$
  loop $j$ : 1 to $R_T$
    $R_c(j) = \{R_c(j-1) + Inc(j)\} \% 32$
    $M_1(i, j) = \{T_{bas}(j) + M_1(i-1, j)\} \% C_T$
    $I_1(i.j) = M_1(i, j)$
    loop $m$ : 2 to 8
        $M_m(i, j) = \{T_{aux}(j) + M_{m-1}(i, j)\} \% C_T$
        $I_m(i, j) = M_m(i, j) + 2^n \bullet j_{(flipped)}$
    end : loop $m$
  end : loop $j$
end : loop $i$
```

and N=12282 respectively. The new lookup tables (T_{bas}) and the corresponding increment step for row counter are given in Table 2. After skipping the un-necessary computations which appear on regular intervals we end up with total computations of 1152 and 12288 for N = 1146 and 12282 respectively. There are only six extra computations needed to get the whole range of addresses and at the same time we get the block sizes which exactly divide by 8.

While working for the parallel address generation, the computation intensive part is to compute M in the algorithm. Using the straight forward method generation of 8 parallel M values requires up to 8 multipliers. We present here a novel approach of using recursive computation for computation of M. After getting the parameter M for the first SISO, the rest of the M values for parallel addresses can be computed recursively over the whole column size as shown in Figure 8. The lookup table entries required to produce the correct M values for parallel addresses at each row instant are named as T_{aux} and are given in the Table 2. The modified algorithm supporting generation of 8 parallel addresses (I_m) is given as Algorithm 1 and the hardware for computing parallel addresses is shown in Figure 14.

5 Parallel Interleaver for 3GPP-LTE

The channel coding in LTE involves Turbo Code with an internal interleaver which is based on quadratic permutation polynomial (QPP). QPP interleavers have very compact

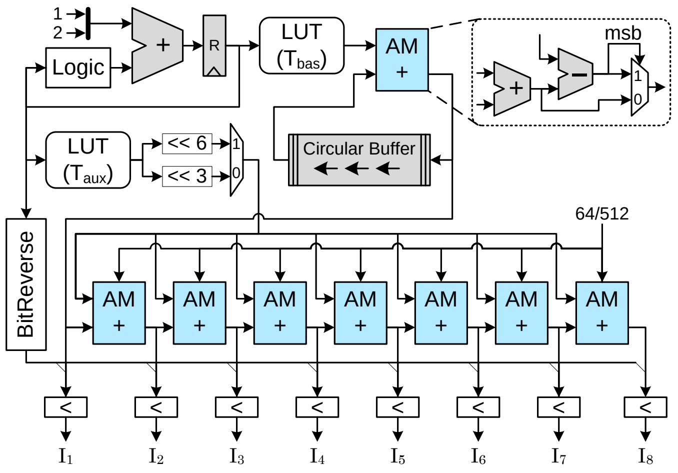

Figure 14: HW supporting 8 parallel interleaver addresses for DVB-SH.

representation methodology and also inhibit a structure that allows the easy analysis for its properties.

The internal interleaver for turbo code is specified by the following quadratic permutation polynomial:

$$I_{(x)} = \left(f_1.x + f_2.x^2\right) \% N \qquad (7)$$

Here $x = 0,1,2,\ldots(N-1)$, where N is block size. This polynomial provides deterministic interleaver behavior for different block sizes with appropriate values of f_1 and f_2. Direct implementation of the permutation polynomial given in eq. (7) is in-efficient due to multiplications, modulo function and bit growth problem. The simplified hardware solution is to use recursive approach described in the following sub-section.

5.1 Basic Interleaver

Eq. (7) can be re-written for recursive computation as:

$$I_{(x+1)} = \left(I_{(x)} + g_{(x)}\right) \% N \qquad (8)$$

Where $g_{(x)} = \left(f_1 + f_2 + 2.f_2.x\right) \% N$, which can also be computed recursively as $g_{(x+1)} = \left(g_{(x)} + 2.f_2\right) \% N$. The two recursive terms $I_{(x+1)}$ and $g_{(x+1)}$ are easy to compute and they provide the basic interleaver functionality for LTE.

5.2 HW for Parallel Interleaver in UMTS-LTE

The interleaver used in LTE is special in the sense that it is conflict free inherently when parallel interleaving is required, thus providing ease of implementation for parallel turbo decoding. Generation of parallel interleaver addresses can be achieved with the replication of the hardware shown in Figure 15 and getting support from a LUT providing the starting values. In this case a total of 32 additions are needed; however,

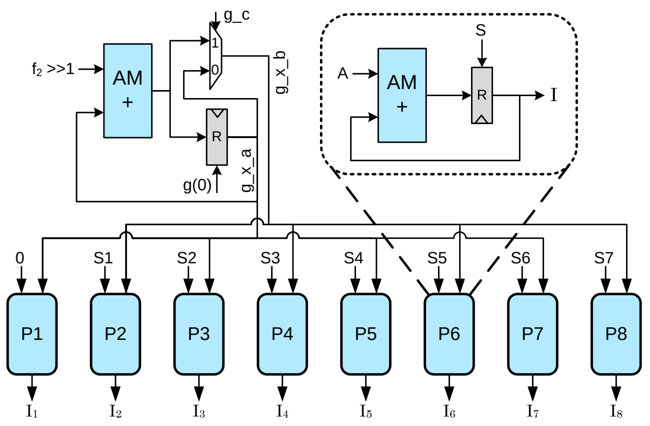

Figure 15: HW for parallel interleaving in 3GPP-LTE.

to achieve a low cost solution we present here the hardware with first part being reused for multiple stages. This optimized hardware uses 18 additions in total to generate 8 parallel interleaver addresses, thus saving 14 additions.

6 Parallel Interleaver for WiMAX

IEEE-802.16e standard [6] called WiMAX provides the mobility features to broadband technology with higher transmission speed i.e. upto 70 Mb/sec. Convolutional turbo coding (CTC) is adopted in the FEC block, which requires an internal interleaver.

6.1 CTC Interleaver

CTC, also termed as duo-binary turbo codes can offer many advantages like performance, over the classical single-binary turbo codes. Work in [19] uses some algorithmic simplifications to implement the CTC interleaver, which makes it possible to compute the interleaver address recursively. The interleaver in the duo-binary turbo codes works on pairs of bits. Presence of CTC interleaver guarantees that the two output parts, systematic output and parity output become completely un-correlated during transmission. Parameters to define the interleaver function as described in [6] 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*
  $$u_0 = \left[(A_0, B_0),(A_1, B_1),(A_2, B_2),....(A_{N-1}, B_{N-1})\right]$$
  for $x = 0.....N-1,$ *if* $(i\%2)=1$ *then* $(A_i, B_i) = (B_i, A_i)$.
  The new sequence is
  $$u_1 = \left[(A_0, B_0),(B_1, A_1),(A_3, B_3),....(B_{N-1}, A_{N-1})\right],$$
- *Step 2: The function* $I_{(x)}$ *providing the mapping address is defined by a set of 4 expressions with a switch selection:*
  for $x = 0......N-1,$ *switch* $(x\%4)$

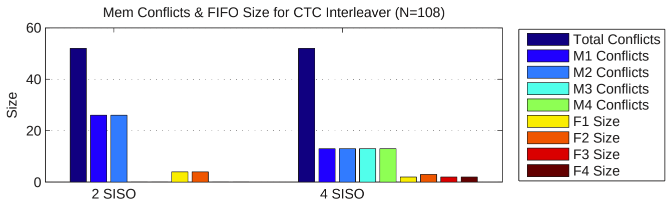

Figure 16: Conflict count and FIFO requirement for N=108 (WiMAX).

Algorithm 2: Proposed Parallel Address Generation Algorithm for WiMAX.

```
Initialization:
$N_{sub\_blk} = \frac{N}{No.of\ SISO}$
$init\ \beta_{(0)} = 0\ \ and\ \ I_{(0)} = 0$
Execution:
$loop\ \ j : 1\ to\ N_{sub\_blk}$
        $\beta_{(j)} = (\beta_{(j-1)} + P_0)\%N$
        $I^1_{(x)} = (\beta_{(j)} + Q_x)\%N$
        $loop\ \ m : 2\ to\ 8$
                $I^m_{(x)} = (I^{m-1}_{(x)} + N_{sub\_blk})\%N$
        $end : loop\ m$
$end : loop\ j$
```

$$\textit{case 0: } I_{(x\%4=0)} = (P_0.x+1)\%N$$
$$\textit{case 1: } I_{(x\%4=1)} = (P_0.x+1+\tfrac{N}{2}+P_1)\%N$$
$$\textit{case 2: } I_{(x\%4=2)} = (P_0.x+1+P_2)\%N$$
$$\textit{case 3: } I_{(x\%4=3)} = (P_0.x+1+\tfrac{N}{2}+P_3)\%N$$

Combining the four equations $I_{(x)}$ becomes:

$$I_{(x)} = (\beta_x + Q_x)\%N \tag{9}$$

Where β_x can be computed using recursion i.e. $\beta_{(x+1)} = (\beta_x + P_0)\%N$ by initializing $\beta_0 = 0$ and Q_x is given by:

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

As range of β_x and Q_x is less then N, thus I_x can be computed by using additions only.

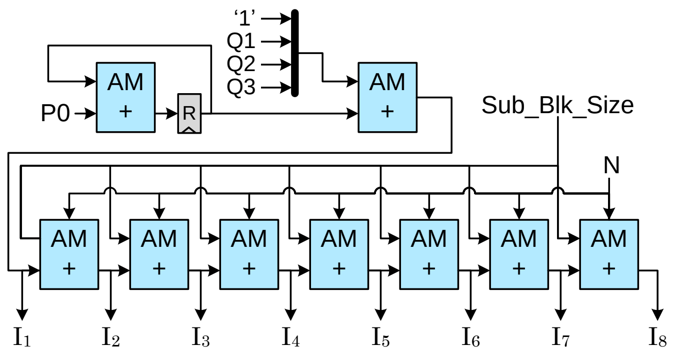

Figure 17: Hardware for parallel interleaving in WiMAX.

Table 3: Permutations for Correct Memory Mapping in WiMAX.

Parallel Address Generation Sequence	I1	I2	I3	I4	I5	I6	I7	I8
Permutation for N = 480 & N = 960	1	6	3	8	5	2	7	4

6.2 HW for Parallel Interleaver in WiMAX

The maximum block size in the normal transmission scheme (i.e. non H-ARQ) is 240, thus in order to achieve sufficient bandwidth, up to 4 parallel SISO processors are enough. On the other hand H-ARQ involves bigger block sizes (max. of 2400) thus up to 8 parallel SISO processors are useful to reduce the latency and enhance the overall bandwidth. While generating parallel interleaver addresses the only case which differs in terms of memory conflict is the block size N=108 (corresponding to QPSK rate:3/4 or 64-QAM rate:3/4). All the other block sizes are supportive to parallelism and are inherently conflict free. The number of conflicts and optimal size of FIFO registers required to handle the conflicts are shown in Figure 16.

The generation of parallel interleavers other then the basic interleaver can be done by successive computations based on the result from the predecessor as given in Algorithm 2. The hardware for the generation of up to 8 parallel interleaved addresses for WiMAX is shown in Figure 17. While generating 8 parallel addresses for the block sizes supported by H-ARQ, some permutation among the generated address is required for selected block sizes (N=480 & N=960) as shown in Table 3. All the other block sizes have the straight one-to-one mapping on different memories.

7 Unified Parallel Interleaver Architecture

The interleaver parallelism for individual standards has been explored in detail in the earlier sections and the number of parallel SISO processors to be supported is summarized in Table 4. The main focus of the work has been to adopt a methodology which results in common computing elements, thus preparing grounds for efficient hardware multiplexing. As a result we reach to the conclusion that an accumulation followed by modulo logic (*acc_mod*) is the common computing element. Therefore forming an array of *acc_mod* with re-configurability for different combinations can serve as a main part in the complete computing core along with an auxiliary part consisting of a multiplication and comparator. As a mandatory part for some of the applications a circular buffer of size 24 is also needed to be incorporated with *acc_mod_array*. The address computation follows the conflict resolution which mainly comprises of multiplexers and shift registers. The complete hardware block diagram for the re-configurable parallel address generation is shown in Figure 18.

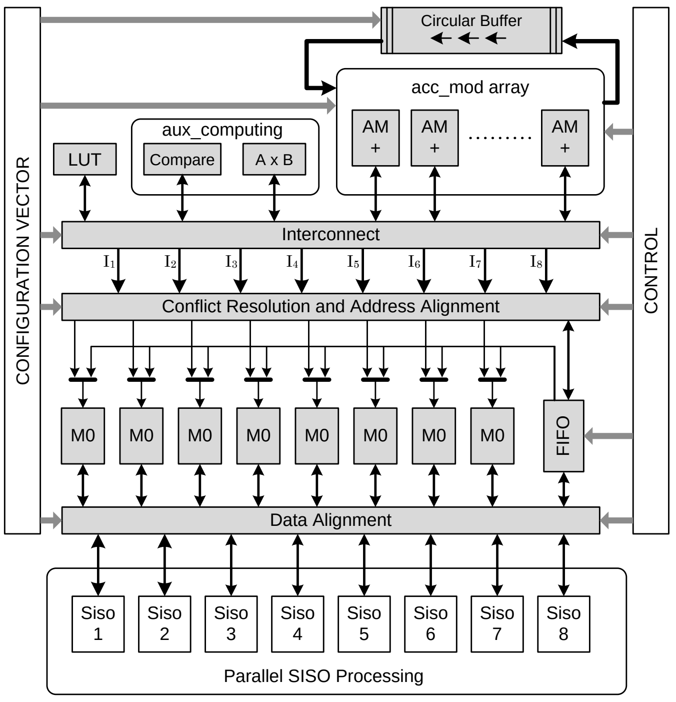

Figure 18: Unified parallel interleaver architecture.

The second part of the conflict management is to apply FIFO register bank dedicated for each memory. The size of the FIFO register bank could have been very large but we reduced it using the progressive writes during a situation of conflict in other memories. Covering all the standards, the FIFO size requirement after applying progressive writes for different memories is given in Table 5. The role of controller is very important to achieve this goal as it checks continuously the empty slots for the corresponding memory. If some other memory has the conflict at certain time instant so that the corresponding memory is free, the controller initiates the left over writes for this memory. The other main tasks handled by controller are to control the sequence of operations during pre-processing and execution phase. The pre-processing includes

Table 4: Interleaver Parallelism in Different Standards.

Parameter	HSPA+	DVB-SH	3GPP-LTE	WiMAX
Block Size (Max)	5114	12282	6144	2400
Data Rate (Mbps)	43.2	50	100	70
Parallel SISO Support	2	2, 4 or 8	2, 4 or 8	2, 4 or 8

Table 5: FIFO Size Requirement for Different Memories.

Scheme	Parameter	M1	M2	M2	M4	M5	M6	M7	M8
HSPA+	Conflicts	0	4	39	0	22	17	--	--
	FIFO Size	0	4	12	0	1	1	--	--
DVB-SH	Conflicts	3072	3072	2304	2304	1344	1344	1344	1344
	FIFO Size	2	2	1	1	1	1	1	1
LTE	Conflicts	0	0	0	0	0	0	0	0
	FIFO Size	0	0	0	0	0	0	0	0
WiMAX	Conflicts	26	26	13	13	--	--	--	--
	FIFO Size	4	4	2	2	--	--	--	--
Final FIFO Size		4	4	12	2	1	1	1	1

Table 6: Summary of Implementation Results.

Parameter	Value with 6K Mem	Value with 12K Mem
Total Area	84330 μm^2	123810 μm^2
Total Power Consumption	12.04 mW	12.65 mW
Memory Configuration	768 x 5b x 8	1536 x 5b x 8
Memory Area	66240 μm^2 (78.5 %)	105720 μm^2 (85.3 %)
Memory Power	11.48 mW (95.3 %)	12.09 mW (95.5 %)
AGU Area	18090 μm^2	
AGU Power	0.56 mW	
Clock Rate	200 MHz	

computation of necessary parameters while changing the standard or block size. During the execution phase the controller keeps track of block size employing row and column counters, thus providing the block synchronization required for each type of interleaver implementation.

There are 8 memories (M1 – M8) being considered as part of interleaver, each having a size of 1536 x 5b to cover the complete range of block sizes for all standards. The memory selection is mainly made by using MSB part of the generated address or though comparison.

8 Implementation Results

By exploiting the hardware re-use for different implementations the final architecture shown in Figure 18 achieves the objective of being low cost. The design is specified in Verilog HDL and synthesized using 65nm CMOS technology. The synthesis results for two cases i.e. with 6K memory and 12K memory are summarized in Table 6. The 6K memory size covers all the cases except one i.e. N=12282 for DVB-SH, thus if specifically this block size is required then

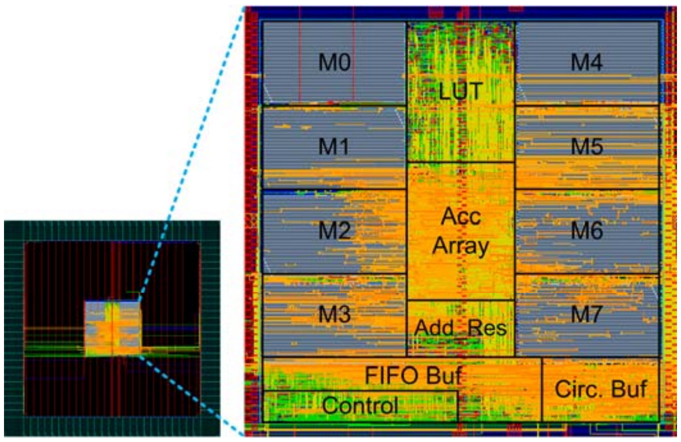

Figure 19: Layout snapshot of proposed unified interleaver.

bigger memory can be used. The chip core layout is shown in Figure 19 and it utilizes $0.085 mm^2$ area in total for 6K memory case. It can operate at a frequency of 200 MHz and consumes $12 mW$ power in total where most of the power utilization is from memories. The address generation hardware for re-configurable parallel interleaver is very much silicon efficient and low power consuming which makes it a good choice for all future implementations targeting the multi-mode operation.

9 Conclusion

In this paper a multi-mode parallel interleaver architecture targeting parallel SISO decoding for different standards has been presented. The design is low cost and supports high frequency at low power. The conflicts, occurring while generating parallel addresses, have been managed by incorporating different schemes. The interleaver address generation hardware for different standards has been modified in the way that efficient hardware multiplexing can be achieved. The functionally of parallel turbo decoder for different standards is more or less same, but parallel interleaving appears to be the bottleneck to reach to a unified version of parallel turbo decoding. The proposed architecture plays a vital role to achieve unified parallel turbo decoding with multi-standard support by using the existing architectures.

References

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

[2] 3GPP, “Technical Specification Group Radio Access Network; Multiplexing and Channel Coding (FDD) (25.212 V5.9.0),” June 2004.

[3] 3GPP, "Technical Specification Group Radio Access Network; Multiplexing and Channel Coding (FDD) (25.212 V8.4.0)," Dec. 2008.

[4] ETSI EN 302-583 V1.1.1, "Digital Video Broadcasting (DVB); Framing Structure, Channel Coding and Modulation for Satellite Services to Handheld Devices (SH) below 3 GHz." March. 2008.

[5] 3GPP-LTE, "Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding," Release 8, 3GPP TS 36.212 v8.0.0, (2007-09).

[6] 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".

[7] Y. Zhang and K.K. Parhi, "Parallel turbo decoding," Proceedings of ISCAS, May 2004, vol. 2, pp. II-509-512.

[8] Y. C. Lu and E.H. Lu, "A parallel decoder design for low latency turbo decoding," Proceedings of ICICIC, Sept. 2007, pp. 386-389.

[9] Y. Sun, Y. Zhu, M. Goel and J. R. Cavallaro, "Configurable and scalable high throughput turbo decoder architecture for multiple 4G wireless standards," Proceedings of ASAP 2008, pp. 209 – 214.

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

[11] R. Dobkin, M. Peleg and R. Ginosar, "Parallel interleaver design and VLSI architecture for low-latency MAP turbo decoders," IEEE Transaction on VLSI Systems, vol. 13, no. 4, April 2005, pp. 427 – 438.

[12] A. Giulietti, L. van der Perre, M. Strum, "Parallel turbo coding interleavers: avoiding collisions in accesses to storage elements," IEE Electronic Letters, Vol. 38, No. 5, pp. 232-234, Feb. 2002.

[13] J. Kwak, S-M. Park, S-S. Yoon and K. Lee, "Implementation of a parallel turbo decoder with dividable interleaver," Proceedings of ISCAS, May 2003, vol. 2, pp. 65-68.

[14] M. Shin and I.-C. Park, "Processor-based turbo interleaver for multiple third-generation wireless standards," IEEE Communication Letters, vol. 7, no. 5, May 2003, pp. 210 – 212.

[15] P. Ampadu and K. Kornegay, "An efficient hardware interleaver for 3G turbo decoding," RAWCON, August 2003, pp. 199 – 201.

[16] R. Asghar and D. Liu, "Very low cost configurable hardware interleaver for 3G turbo decoding," Proceedings of ICTTA, April 2008, pp. 1 – 5.

[17] R. Asghar and D. Liu, "Dual standard re-configurable hardware interleaver for turbo decoding," Proceedings of ISWPC, May 2008, pp. 768 – 772.

[18] Z. Wang and Q. Li, "Very low-complexity hardware interleaver for turbo decoding," IEEE Transaction on Circuits and System – II: vol. 54, no. 7, July 2007, pp. 636 – 640.

[19] R. Asghar and D. Liu, "Low complexity multi mode interleaver core for WiMAX with support for convolutional interleaving," Journal of Elec., Comm. and Computer Engineering, vol. 3, no. 1, 2009, pp. 20 – 29.

[20] G. R. Blakley, "A computer algorithm for calculating the product A*B mod M," IEEE Transaction on Computers, vol.C-32, No.5, May 1983, pp.497–500.

[21] R. Asghar, D. Wu, J. Eilert and D. Liu, "Memory conflict analysis and interleaver design for parallel turbo decoding supporting HSPA evolution," Accepted for publication in 12th Euromicro DSD-2009, August 2009.

[22] F. Speziali and J. Zory, "Scalable and area efficient concurrent interleaver for high throughput turbo-decoders," Proceedings of Euromicro DSD-2004, pp. 334 - 341.

[23] M. Martna, M. Nicola and G. Masera, "Hardware design of a low complexity, parallel interleaver for WiMAX Duo-Binary turbo decoding," IEEE Communication Letters, vol. 12, no. 11, pp. 846 - 848, November 2008.

Rizwan Asghar is working on a Ph.D. degree at the Department of Electrical Engineering, Linköping University, Sweden. He received the 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, Islamabad, affiliated with U.E.T. Texila, Pakistan. His research activity is mainly focused on flexible and re-configurable FEC sub-systems for baseband processors.

Di Wu received his M.Sc. in electrical engineering both from Beijing University of Posts and Telecoms and Linköpings Universitet, Linköping, Sweden, in 2003 and 2005 respectively. He is currently a Ph.D. candidate at the Division of Computer Engineering at the Department of Electrical Engineering at the same university. His research interests include application specific processor design and fast prototyping of wireless systems.

Johan Eilert received his M.Sc. in computer science and engineering from Linköpings Universitet, Linköping, Sweden, in 2003. He is currently a Ph.D. candidate at the Division of Computer Engineering at the Department of Electrical Engineering at the same university. His research interests include application specific processors for multimedia signal processing and MIMO baseband signal processing.

Dake Liu is Professor and the Director of Computer Engineering Division at the Department of Electrical Engineering of Linköping University, Sweden. He got technology doctor degree from Linköping University Sweden in 1995. He is IEEE senior member. Dake published more than 100 papers on journals and international conferences and holds 5 US patents. Dake's research interests are high-performance low-power ASIP (application specific instruction set processors), integration of on-chip multi-processors for communications, and media digital signal processing. Dake has experiences also in design of communication systems and Radio frequency CMOS integrated circuits. Dake Liu is the co-founder and CTO of FreehandDSP AB, Stockholm Sweden. FreehadDSP was acquired by VIA technologies (http://www.viatech.com/en/index.jsp) in 2001. Dake Liu is currently the co-founder and CTO of Coresonic AB, (http://www.coresonic.com/) Linköping Sweden.