Highly Scalable On-the-Fly Interleaved Address Generation for UMTS/HSPA+ Parallel Turbo Decoder

Aida Vosoughi[1], Guohui Wang[1], Hao Shen[1], Joseph R. Cavallaro[1], and Yuanbin Guo[2]
[1] Department of Electrical and Computer Engineering, Rice University, Houston, Texas, USA
[2] Wireless R&D, US Research Center, Futurewei Technologies, Plano, Texas, USA
{vosoughi, wgh, hs9, cavallar}@rice.edu, yuanbinguo@huawei.com

***Abstract*— High throughput parallel interleaver design is a major challenge in designing parallel turbo decoders that conform to high data rate requirements of advanced standards such as HSPA+. The hardware complexity of the HSPA+ interleaver makes it difficult to scale to high degrees of parallelism. We propose a novel algorithm and architecture for on-the-fly parallel interleaved address generation in UMTS/HSPA+ standard that is highly scalable. Our proposed algorithm generates an interleaved memory address from an original input address without building the complete interleaving pattern or storing it; the generated interleaved address can be used directly for interleaved writing to memory blocks. We use an extended Euclidean algorithm for modular multiplicative inversion as a step towards reversed intra-row permutations in UMTS/HSPA+ standard. As a result, we can determine interleaved addresses from original addresses. We also propose an efficient and scalable hardware architecture for our method. Our design generates 32 interleaved addresses in one cycle and satisfies the data rate requirement of 672 Mbps in HSPA+ while the silicon area and frequency is improved compared to recent related works.**

Keywords—Turbo decoder; Interleaved Address Generation; Interleaved writing; UMTS; HSPA+; Parallel turbo decoder

I. INTRODUCTION

Turbo code has been adopted as the forward error correction channel code in many of the existing and new wireless communication standards such as Universal Mobile Telecommunications System (UMTS), High-Speed Packet Access Evolution (HSPA+) and Long Term Evolution (LTE). The most recent HSPA+ standard [1] supports up to 337.5 Mbps in downlink, and up to 672 Mbps has been proposed for the future 3GPP extension [2]. To keep up with the ever-growing data rate requirements in the standards, parallel and high-throughput turbo decoders have attracted a lot of interest recently [3]--[8]. Turbo decoding consists of a sequence of iterations. Each iteration includes two Soft Input Soft Output (SISO) decoding half-iterations and an interleaving step between two half-iterations. An interleaver permutes bits in a pseudo-random fashion and hence makes error distribution more uniform; this effect increases the performance of the channel code. In a parallel turbo decoder, the block of bits are divided into multiple sub-blocks and are fed to multiple parallel SISO decoders; therefore, parallel interleaver blocks are essential in a parallel turbo decoder.

A naïve approach for implementing an interleaver is to use lookup tables (LUT) to store all of the possible interleaving patterns in a memory. The excessive memory requirement of this approach makes it unattractive for highly parallel turbo decoders. Therefore, on-the-fly interleaving has been preferred to the LUT approach in almost all of the recent turbo decoder designs.

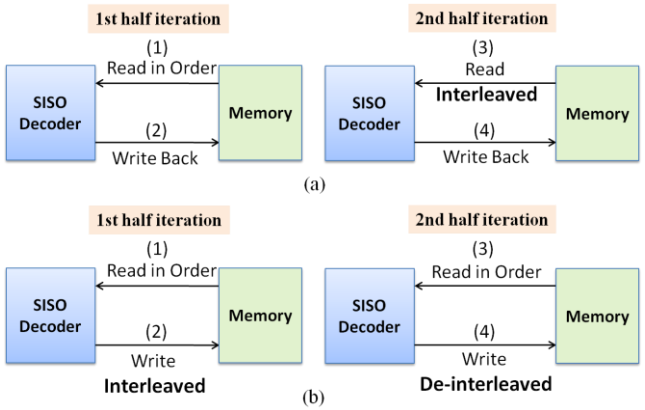

Fig. 1. Memory access scheduling in turbo decoding. (a) Conventional way. Write back means the data is written back to the same address from which it is read (b) Proposed scheduling for high-throughput parallel decoding [3]. Numbers in parentheses show scheduling steps.

Often, turbo decoders use memory blocks to store the intermediate values in between decoding iterations. Therefore, the read and write memory addresses are interleaved. Traditionally, each parallel SISO decoder in a parallel turbo decoder performs in-order read in the first half-iteration and interleaved read in the second half iteration. Also, in both half iterations data is written back to the same address from which it is read from [7], [9] (see Figure 1(a)). However, memory contention (i.e. when multiple reads/writes are performed by multiple SISO decoders on the same memory block at the same time) is a key obstacle in achieving high throughput turbo decoders. We have shown in recent work [3] that a new scheduling consisting of in-order reading in both half-iterations and interleaved/de-interleaved writing in first/second half-iterations eliminates reading conflicts and therefore simplifies the hardware and improves the turbo decoder throughput (please refer to [3] for details). Figure 1(b) shows the new turbo decoding memory access scheduling introduced in [3]. In such a configuration, writing to memory is done in an interleaved fashion; that is, data is read (in order) from an (original) address a, but after processing, the corresponding result must be written into the address calculated as interleaved(a) (or I(a)); therefore, we need to generate the interleaved addresses from the original addresses.

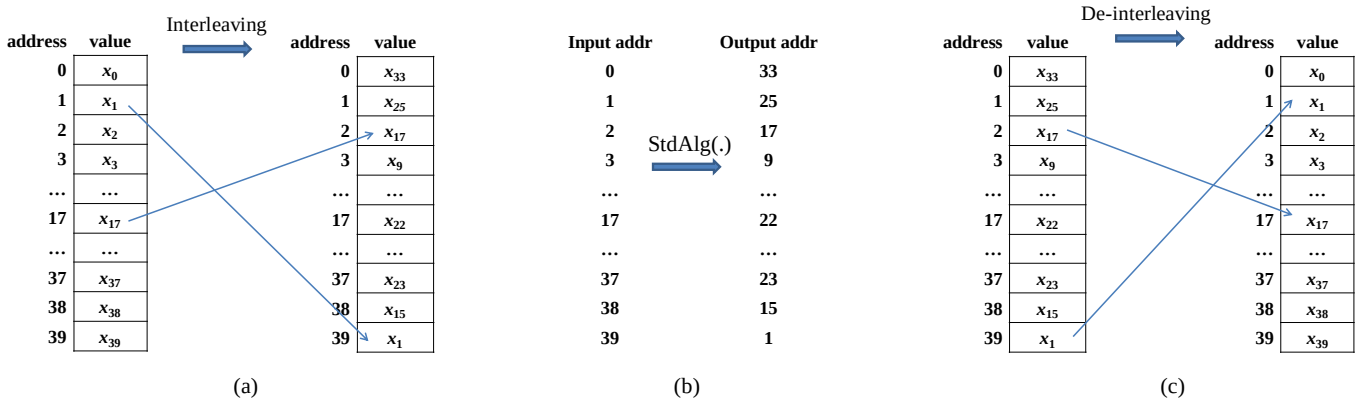

Fig. 2. (a) UMTS/HSPA+ memory interleaving process example for minimum block size (K=40). As an example, for interleaved writing, we need the mapping I(1)=39 and I(17)=2, where I(.) is the interleaved address generation function that takes original address and outputs interleaved address. (b) UMTS/HSPA+ interleaver algorithm mapping for block size 40, StdAlg(.) is the UMTS/HSPA+ standard interleaving algorithm. (c) UMTS/HSPA+ memory de-interleaving process for minimum block size (K=40). For de-interleaved writing, we need D(2)=17 and D(39)=1, where D(.) is the de-interleaved address generation function.

The previously proposed on-the-fly UMTS interleavers [5], [6], [9]--[12] are designed to be used in parallel turbo decoders with conventional scheduling, where reading is done in an interleaved fashion. In these methods, when the *b-th* interleaved data must be read, the original address of b is computed and used for reading. Therefore, the original addresses are generated from interleaved addresses based on the closed form permutation expressions given in the UMTS standard [1]. However, these interleaving methods cannot be used in the new highly-parallel high throughput turbo decoders that use a memory access scheduling based on the proposed method in [3], because they are unable to generate interleaved address from an original address.

In the new proposed scheduling that minimizes memory contention [3] (shown in Figure 1(b)), reading is in-order and writing to memory is done in interleaved (or de-interleaved in second half-iteration) order. For interleaved writing, we need to perform the reversed form of standard permutations [1] to derive interleaved addresses from original addresses. In this paper, we propose a novel method to perform interleaved writing without wasting resources to produce the whole interleaving pattern (sequence of addresses). ***Our contributions are:*** **(1)** We propose an algorithm to generate an interleaved address I(a) from any arbitrary original input address a which can be used efficiently for interleaved writing to memory. The output address is generated on-the-fly without the need for a sort or search through a pattern; moreover, our method does not require storing patterns in a memory. **(2)** We propose and implement an efficient interleaved address generation hardware architecture for the introduced algorithm. We report the resource usage and timing results of our implementation in ASIC library TSMC 65nm. **(3)** We show that our proposed interleaving method and architecture are easily and efficiently scalable for a highly parallel turbo decoder. Resource reuse makes this extension affordable.

II. UMTS/HSPA+ Standard Interleaver

The turbo code interleaver algorithm for UMTS/HSPA+ is described in the standard [1]. In a nutshell, the UMTS/HSPA+ turbo interleaver works as follows: First, the block of input bits is padded by dummy bits and is shaped into a rectangular matrix. The rectangular matrix is then permuted with intra-row and inter-row pseudo-random permutations. Finally, bits are read from the matrix and pruned back to the original block length. The UMTS standard interleaver consists of several computationally costly steps such as multiplication and modulo operation. Moreover, UMTS/HSPA+ interleaver has not originally been designed for parallel decoding and is not contention-free [4]. These complications have made UMTS/HSPA+ interleaver attractive for research in the last decade. As explained previously, turbo decoders are often implemented such that the intermediate values are stored in a memory. Figure 2(a) shows the memory interleaving process in UMTS/HSPA+ standard for the minimum block size of 40. The input bit sequence to the interleaver is denoted by $x_0, x_1, x_2, \ldots, x_{39}$. For interleaved writing, our desired on-the-fly interleaved address generator must take the current (original) memory address of a value and output the corresponding interleaved address (i.e. I($address$)). For example, as shown in Figure 2(a), the interleaved address generation unit must produce I(1)=39 and I(17)=2. However, the UMTS/HSPA+ interleaving algorithm generates a pattern like the one shown in Figure 2(b). This mapping, denoted by function StdAlg(.), is not equivalent to our desired mapping. For instance, StdAlg(1)=25 and StdAlg(17)=22. Therefore, the original UMTS/HSPA+ interleaving algorithm cannot be directly applied for interleaved writing where we need to generate an interleaved address from its original position.

In fact, the standard algorithm can be used, as is, for de-interleaved address generation (for de-interleaved writing). Figure 2(c) shows the standard de-interleaving process for K=40 (function D(.)). Note that the desired mapping for de-interleaving, shown in Figure 2(c), is exactly equal to the mapping that the UMTS/HSPA+ standard algorithm generates (Figure 2(b)). For instance, D(2)=StdAlg(2)=17 and D(39)=StdAlg(39)=1.

In the next section, we propose a method that performs reversed permutations to generate the desired function that takes original addresses as input and generates the interleaved addresses as output. Although the described algorithm here is specific to UMTS/HSPA+ standard interleaver, the same methodology for generating reversed permutations can be

adopted in other communication standards that make use of a pseudo-random interleaver for turbo code [13].

III. Proposed Algorithm for UMTS/HSPA+ Interleaved Address Generation

In this section, we first describe the steps of our proposed algorithm for on-the-fly interleaved address generation based on the UMTS/HSPA+ standard interleaving algorithm [1] and then discuss the main steps in more detail. Please refer to UMTS standard [1] for details of the algorithm. Our proposed method extends the standard algorithm to perform the reversed form of the standard intra- and inter-row permutations. As a result, we generate an interleaved address from an input address. The notations are adapted from the standard and are listed below.

$\boldsymbol{a}$: Input to the algorithm (original address to be interleaved), $\boldsymbol{a'}$: Output of the algorithm (interleaved address), $\boldsymbol{K}$: Block size, $\boldsymbol{R}$: Number of rows of rectangular matrix, $\boldsymbol{C}$: Number of columns of rectangular matrix, $\boldsymbol{I}$: Row index of a in the original rectangular matrix, $\boldsymbol{J}$: Column index of a in the original rectangular matrix, $\boldsymbol{I'}$: Row index of a'in the interleaved rectangular matrix, $\boldsymbol{J'}$: Column index of a' in the interleaved rectangular matrix, $\boldsymbol{p}$: Prime number, $\boldsymbol{v}$: Primitive root, $\langle \boldsymbol{s(j)} \rangle_{\boldsymbol{j\in\{0,1,\dots,p-2\}}}$: Base sequence for intra-row permutation, $\langle \boldsymbol{s\_inv(j)} \rangle_{\boldsymbol{j\in\{0,1,\dots,p-2\}}}$: Reverse of base sequence for intra-row permutation, $\langle \boldsymbol{q_i} \rangle$: Minimum prime integers, $\langle \boldsymbol{r_i} \rangle$: Permuted prime integers, $\langle \boldsymbol{m_i} \rangle$: Modular multiplicative inverse of $\langle r_i \rangle$, $\langle \boldsymbol{T(i)} \rangle_{\boldsymbol{i\in\{0,1,\dots,R-1\}}}$: Inter-row permutation pattern, $\langle \boldsymbol{T\_inv(i)} \rangle_{\boldsymbol{i\in\{0,1,\dots,R-1\}}}$: Reverse of inter-row permutation pattern, $\langle \boldsymbol{d(j)} \rangle_{\boldsymbol{j\in\{0,1,\dots,C-1\}}}$: The number of dummy bits in columns with index smaller than j in the interleaved matrix, $\boldsymbol{dummy\_cols\_set}$: The set of column indexes of dummy bits after interleaving, $\boldsymbol{dummy\_row}$: The row index of dummy bits after interleaving (note that the entire dummy values reside in a same row after interleaving).

The steps of the proposed algorithm for interleaved address generation are listed below. Steps (1)--(4), (6), and (7) are equivalent to the standard algorithm [1]. Steps (5), (8), (10), and (11) include the main contribution of our proposed algorithm (these steps are **boldface** in the text). These steps are designed to reverse the standard intra-row and inter-row permutations. This is needed to generate interleaved address from original address.

A. Steps of the Proposed Algorithm

(1) Derive the number of rows of the rectangular matrix:

$$R = \begin{cases} 5, & \text{if } (40 \le K \le 159) \\ 10, & \text{if } ((160 \le K \le 200) \text{ or } (481 \le K \le 530)) \\ 20, & \text{otherwise} \end{cases}$$

(2) Determine the prime number to be used in the intra-row permutation, p, and the number of columns, C:

If $(481 \le K \le 530)$ then $p = 53$ and $C = p$, otherwise, find the minimum prime number, p, from prime number table from the standard [1] such that $K \le R \times (p+1)$, and determine C such that,

$C = p-1$; if $K \le R \times (p-1)$;
$C = p$; if $R \times (p-1) < K \le R \times p$;
$C = p+1$; if $R \times p \le K$;

(3) Select the primitive root, v, associated with the selected prime number, p, from standard prime number table.

(4) Construct the intra-row permutation sequence $s(j)$: $s(j) = \left(v \times s(j-1)\right) mod\ p$, for $j = 1,2,\dots,p-2$, and $s(0) = 1$.

(5) Construct the sequence $\boldsymbol{s\_inv(j)}$ from $\boldsymbol{s(j)}$ such that, $\boldsymbol{s\_inv(s(j)) = j}$ for $\boldsymbol{j = 0,1,2,\dots,p-2}$.

(6) Determine the prime integer q_i in the sequence $\langle q_i \rangle_{i\in\{0,1,\dots,R-1\}}$ to be the least prime integer such that $\gcd(q_i, p-1) = 1, q_i > 6, and\ q_i > q_{i-1}$(gcd denotes the greatest common divisor). Assign $q_0 = 1$.

(7) Permute the sequence q_i to make sequence $\langle r_i \rangle_{i\in\{0,1,\dots,R-1\}}$ as follows: $r_{T(i)} = q_i,\ i = 0,1,\dots,R-1$, where $\langle T(i) \rangle$ is one of the four possible inter-row permutation patterns defined in the standard [1] based on R and K.

(8) Construct the modular multiplicative inverse sequence $\langle \boldsymbol{m_i} \rangle_{\boldsymbol{i\in\{0,1,\dots,R-1\}}}$,where $\boldsymbol{m_i = r_i^{-1} mod\ (p-1)}$ for $\boldsymbol{i \in \{0,1,\dots,R-1\}}$. (i.e. $\boldsymbol{m_i}$ is the modular multiplicative inverse of $\boldsymbol{r_i}$ modulo $\boldsymbol{p-1}$).

(9) Determine the row index (I) and the column index (J) of the original input address (a):

$$I = \left\lfloor \frac{a}{C} \right\rfloor, \text{ and } J = a - I \times C$$

(10) Perform the proposed reversed intra-row permutation as shown in Figure 3.

```
if (C = p)
  if (J = 0)
    J' = p − 1;
  else
    J' = m_I × s_inv(J) mod p − 1;
if (C = p + 1)
  if (K = R × C) and (I = R − 1) and (J = 1)
    J' = p;
  else if (K = R × C) and (I = R − 1) and (J = p)
    J' = 0;
  else
    if (J = 0)
      J' = p − 1;
    else if (J = p)
      J' = p;
    else
      J' = m_I × s_inv(J) mod p − 1;
if (C = p − 1)
  J' = m_I × s_inv(J + 1) mod p − 1;
```

Fig. 3. Reversed intra-row permutation for Step (10)

(11) Perform the inter-row permutation: $\boldsymbol{I' = T\_inv(I)}$. $\boldsymbol{T\_inv}$ is the reverse of the standard inter-row permutation pattern i.e. $\boldsymbol{T\_inv(T(i)) = i}$, for $\boldsymbol{i \in \{0,1,\dots,R-1\}}$).

(12) Derive the output interleaved address $a' = (J' \times R) + I' - d(J')$.

(13) Exception: if $J' \in dummy\_cols\_set$ and $I' > dummy\_row$, then decrement by 1: $a' = a' - 1$.

B. Description of Main Interleaved Address Generation Algorithm Steps

In the standard UMTS/HSPA+ interleaving algorithm, the inter-row permutation is performed based on the pattern $\langle T(i) \rangle_{i \in \{0,1,\ldots,R-1\}}$, where $T(i)$ is the original row position of the i-th permuted row. In contrast, our proposed method uses T_inv sequences to derive the permuted row from the original row. The UMTS standard intra-row permutation works as the following pseudo-code describes [1]:

if $(\boldsymbol{C = p})$
  $U_i(j) = s((j \times r_i)\ mod\ (p-1)); U_i(p-1) = 0;$
if $(\boldsymbol{C = p+1})$
  $U_i(j) = s((j \times r_i)\ mod\ (p-1)); U_i(p-1) = 0; U_i(p) = p;$
  if $(K = R \times C)$
    Exchange $U_{R-1}(0)$ and $U_{R-1}(p)$;
if $(\boldsymbol{C = p-1})$
  $U_i(j) = s\big((j \times r_i)\ mod\ (p-1)\big) - 1;$

In the closed-form expression in the standard, $U_i(j)$ is the original column position of j-th permuted column of i-th row, where $j = 0, 1, \ldots, p-2$ and $i = 0, 1, \ldots, R-1$. However, our goal is to derive the interleaved column from the original column; therefore, for our proposed algorithm (in step (10)), we go backward in the equations explained above. We use s_inv instead of s and modular multiplicative inverses of $\langle r_i \rangle$ in reversed intra-row permutation (step (10)) for this purpose.

The modular multiplicative inverse of an integer x modulo q is an integer y such that: $x \times y = 1 \pmod{q}$, and it exists if and only if gcd(x,q)=1, where gcd is the greatest common divisor [14]. Since $\langle r_i \rangle$ is a permutation of sequence $\langle q_i \rangle$ and for every element of $\langle q_i \rangle$ we have $\gcd(q_i, p-1) = 1$, therefore the multiplicative inverse modulo $p-1$ exists for every element of $\langle r_i \rangle$. The modular multiplicative inverse of x modulo q can be found using the extended Euclidean algorithm, which besides finding the greatest common divisor of integers x and q as the Euclidean algorithm does, also finds integers A and B that satisfy Bézout's identity: $Ax + Bq = \gcd(x, q)$. When x and q are coprime (i.e. gcd(x,q)=1), A is the modular multiplicative inverse of x modulo q, and B is the modular multiplicative inverse of q modulo x [14]. Figure 4 shows the pseudo-code for iterative extended Euclidean algorithm that we use in our design.

To elaborate how the backward intra-row permutation in step (10) of our proposed algorithm, is derived from the standard intra-row permutation algorithm, we consider the case where $C = p$. According to the standard we have $U_i(j) = s((j \times r_i) \bmod (p-1))$, where $U_i(j)$ is the original column index, i is the original row index, and j is the permuted column index. In the proposed algorithm we denote the original column, original row, and permuted column index, by J, I, and J' respectively. Therefore with replacement we have $J = s((J' \times r_I) \bmod (p-1))$.

Inputs: x, q
Initialization:
  $CurrA$= 0; $LastA$ = 1;
  $CurrB$= 1; $LastB$ = 0;
Iterations:
  while $(q \neq 0)$
    quotient = $x \div q$;
    remainder = x mod q;
    $x = q$; q = remainder;
    $NewA = LastA$ – (quotient × $CurrA$);
    $NewB = LastB$ – (quotient × $CurrB$);
    $LastA = CurrA$; $CurrA = NewA$;
    $LastB = CurrB$; $CurrB = NewB$;
Outputs: $LastA$, $LastB$

Fig. 4. Extended Euclidean algorithm for modular multiplicative inverse. Output *LastA* is the modular multiplicative inverse of input x modulo input q [14].

In this equation, J,I are known and J' is the desired value to be found.
$J = s((J' \times r_I) \bmod (p-1))$,
Performing function $s\_inv(.)$ on both sides, we have:
$s\_inv(J) = s\_inv(s\big((J' \times r_I) \bmod (p-1)\big))$.
Since by definition $s\_inv(s(i)) = i$:
$s\_inv(J) = (J' \times r_I) \bmod (p-1)$.
Multiplying both sides by m_I we get:
$s\_inv(J) \times m_I = (J' \times r_I \times m_I) \bmod (p-1)$.
Since, m_I is the modular multiplicative inverse of r_I mod $(p-1)$:
$s\_inv(J) \times m_I = J' \bmod (p-1)$.
Or, $J' = m_I \times s\_inv(J) \bmod (p-1)$,

which is what we have as the expression in our step (10). The proof of the other cases for intra-row permutation is very similar and the proof for inter-row permutation is trivial.

Steps (12) and (13) consist of calculating the final interleaved address. According to the standard, the final address is derived by counting matrix elements column-wise, from top to bottom and from left to right. Also, the dummy bits must be excluded from counting. This is called pruning in the standard. Therefore, $d(j)$ must be subtracted from the output position. In a special case, the interleaved column index may be the same as one of the dummy bits' column index. In this case, the interleaved address must be decremented by one if the interleaved row is below the row of dummy bits. Note that all of the dummy bits share the same row after interleaving.

IV. Proposed Architecture for On-the-Fly Interleaved Address Generation Unit

We propose a novel hardware architecture for our interleaved address generation algorithm that is scalable for highly parallel interleaving. Figure 5 shows a block diagram of our proposed architecture. Two main processing units are preprocessing and run-time units. The preprocessing unit is responsible for the steps (1)--(8) in the proposed algorithm in the previous section. Run-time unit performs steps (8)--(13). Read only memories (ROMs) hold the parameters that are independent of block size and hence can be computed and fixed offline. The values that are stored in Random Access

Memories (RAMs) are generated and stored by the preprocessing unit and are used by the run-time unit.

The preprocessing unit is activated only when there is a change in the input block size. Whenever K is modified, the preprocessing unit updates the parameters and overwrites the RAMs and registers. The run-time unit, on the other hand, is always active and uses the parameters generated by the preprocessing unit to compute the interleaved address from an input address. The following describes the main blocks of our architecture.

A. Memories

p-v-ROM stores prime numbers (p) and their corresponding primitive roots (v) (Prime numbers table from UMTS standard [1]). **T-ROM** consists of the inter-row permutation patterns, T from the standard [1], while **T-ROM-inv** stores T_inv patterns. **r-ROM** stores r sequences for different p values. **m-RAM** stores the multiplicative modular inverses of $\langle r_i \rangle$ values, which are denoted by $\langle m_i \rangle$. **s-RAM-inv** stores s_inv array computed in preprocessing.

B. Preprocessing Unit

The preprocessing unit takes the block size K ($40 \leq K \leq 5114$ as defined in the standard [1]) as input and generates C (number of columns of the rectangular matrix), R (number of rows of rectangular matrix), p (prime number from prime numbers table from UMTS standard [1]), and v (corresponding primitive root). The modular multiplicative inverse block, shown in the preprocessing unit of Figure 5, performs the iterative extended Euclidean algorithm (shown in Figure 4) to find the inverses ($\langle m_i \rangle$). A detailed block diagram of this block is given in Figure 6. Many $\langle q_i \rangle$ sequences are similar for different values of p; therefore, $\langle q_i \rangle$ sequences can be generated offline, permuted to generate $\langle r_i \rangle$ sequences and grouped and stored in a read-only memory [7]. Offline modulo computation for $\langle q_i \rangle$ sequences saves resources. Finally, the preprocessing unit chooses the right $\langle r_i \rangle$ sequence based on the value of p and generates $\langle m_i \rangle$ as described in step (8) of our proposed algorithm. The preprocessing unit is also responsible for generating the inverse of base sequence $\langle s\_inv(j) \rangle$ and storing them in s-RAM-inv. All of the parameters that are generated by the preprocessing unit are stored in registers and memories and are accessed by both the preprocessing unit and the run-time address generating unit. The preprocessing is done once, and the output values are fixed as long as the block size remains the same. Upon a change in block size, the preprocessing must be redone for the new block size.

C. Run-time Address Generation Unit

The run-time unit uses the values generated by the preprocessing unit to compute the interleaved address from the input address. This unit consists of an address translator unit which takes the original input address and outputs the corresponding row and column index as explained in step (9) in the algorithm. Then the final address is computed after reversed intra-row and inter-row permutations according to steps (10) and (11). Memory access, addition, subtraction, multiplication, and modulo operations are required at run-time. Figure 7 describes the run-time unit.

D. Hardware Synthesis Results

We implemented the proposed architecture with Verilog HDL and synthesized the design with a TSMC 65*nm* standard cell library using Synopsys Design Compiler. Tables I, II, and III summarize resource and memory usage of the proposed hardware implementation of the UMTS/HSPA+ interleaved address generation unit. In Table I, the total cell area and equivalent gate count is reported for preprocessing and run-time units separately.

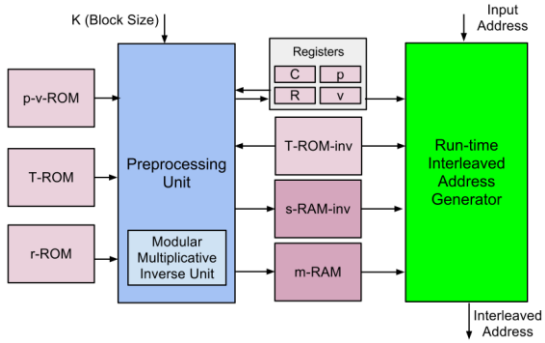

Fig. 5. Proposed architecture for UMTS/HSPA+ interleaved address generation (IAG), non-parallel version.

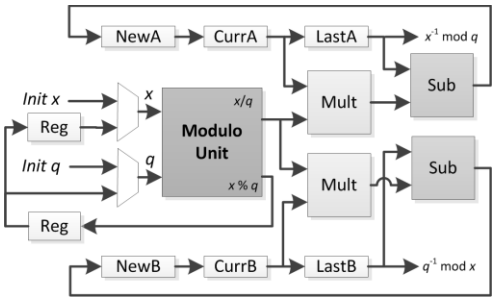

Fig. 6. Modular multiplicative inverse unit based on iterative extended Euclidean algorithm.

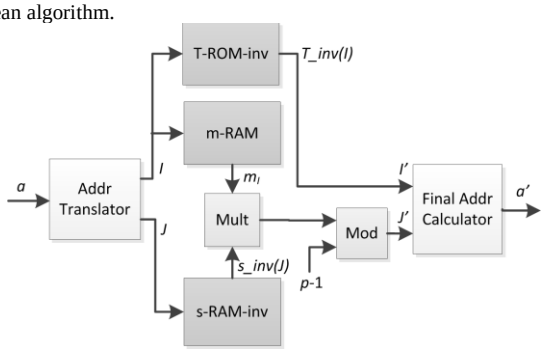

Fig. 7. Interleaved address generation run-time unit

TABLE I. AREA ESTIMATION OF UMTS/HSPA+ INTERLEAVED ADDRESS GENERATION UNIT (TECHNOLOGY: TSMC 65*nm*, FREQ=700MHZ, GATE (2-INPUT NAND) EQUIVALENT = 1.3 μm^2)

Module	Preprocessing unit	Run-time unit	Total
Cell area (μm^2)	3820	3429	7249
Gate count	2938	2638	5576

TABLE II. SIZE OF RANDOM-ACCESS MEMORIES

RAM	s-RAM-inv	m-RAM	Total
Size (Bytes)	256	20	276

TABLE III. SIZE OF READ-ONLY MEMORIES

ROM	T-ROM	T-ROM-inv	p-v-ROM	r-ROM	Total
Size (Bytes)	32	32	84	548	696

V. EXPANSION OF THE PROPOSED ARCHITECTURE FOR A PARALLEL INTERLEAVER

A parallel version of the proposed architecture for UMTS/HSPA+ interleaved address generation unit can be achieved using only one preprocessing unit, multiple copies of (some of) the memory blocks, and multiple copies of the run-time unit. Figure 8 describes the proposed parallel architecture. As shown before in Tables I, II and III, the preprocessing unit and the read-only memories use the largest portion of the total resources required for address generation. Therefore, sharing the preprocessing unit and memories in the parallel architecture saves a large amount of hardware resources. Furthermore, the proposed parallel architecture can be efficiently extended to a dual-mode parallel interleaved/de-interleaved address generation unit where the same preprocessing unit and memories can be used in both modes.

We define the degree of parallelism (PL) as the number of interleaved addresses that are generated in one clock cycle. Here, we assume the decoders are radix-4 XMAP (cross maximum a posteriori probability) SISO decoders [15]. Such a decoder, in each clock cycle reads four extrinsic Log Likelihood Ratio (LLR) values from LLR memory and then outputs four new LLR values, which should be written in LLR memory to the interleaved addresses. Therefore the minimum parallelism that we consider here is 4. For a degree of parallelism equal to PL, a group of PL interleaver run-time units must work in parallel to maximize the throughput. However, as described before, the preprocessing unit and ROMs need not be duplicated as they can be shared between the run-time units. Table IV reports the area and memory size for a parallel UMTS/HSPA+ interleaved address generation unit with different degrees of parallelism (note that the parallelism is not limited to 32). The reported synthesis results are in TSMC 65nm technology, and the target frequency is set to 700MHz. We have also estimated the throughput of the parallel turbo decoder that uses our proposed interleaver design and the contention-free buffer structure proposed in [4]. In throughput estimation, the block size is assumed to be the maximum (5114), and the number of iterations of the turbo decoder is 6 which is a reasonable assumption. Also, a latency of 8 clock cycles is assumed for the turbo decoder to account for possible memory conflict resolution processes. With the above-mentioned assumptions, for PL = 32 the maximum achievable throughput satisfies the throughput requirements of 672 Mbps in HSPA+ standard.

A. Comparisons and Discussions

Figure 9 compares the growth of the area of parallel interleaved address generation hardware with that of SISO decoders for different degrees of parallelism. An estimated area for a radix-4 XMAP decoder is 0.081 mm^2 in the 65nm technology [16]. As is clear from the figure, the area of the proposed interleaved address generator is much smaller than that of SISO decoders; particularly, for higher degrees of parallelism, the area difference is significant.

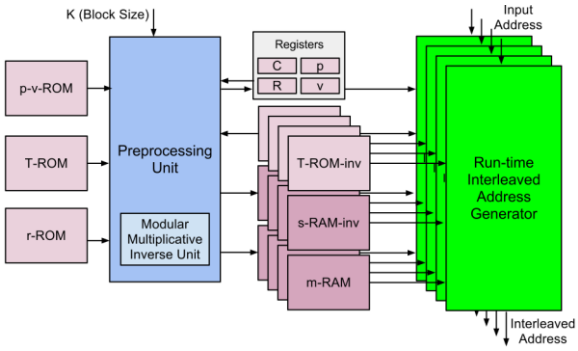

Fig. 8. Parallel high-throughput UMTS/HSPA+ interleaved address generation unit.

TABLE IV. RESOURCE AND THROUGHPUT ESTIMATION FOR PARALLEL UMTS/HSPA+ INTERLEAVED ADDRESS GENERATION UNIT (TECHNOLOGY: TSMC 65nm, FREQ=700MHZ; BLOCK SIZE=5114; NUMBER OF TURBO ITERATIONS=6; TURBO DECODER LATENCY = 8 CLOCK CYCLES.)

Degree of Parallelism (PL)	Area (mm^2)	Memory (KBytes)	Estimated Throughput (Mbps)
4	0.0175	1.89	116
8	0.0312	3.12	230
16	0.0586	5.59	455
32	0.1135	10.52	889

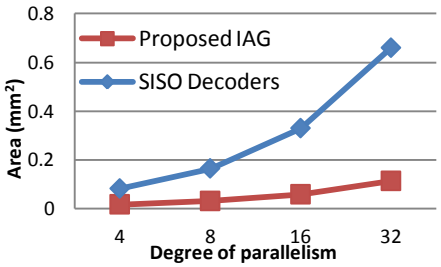

Fig. 9. Comparing area growth of IAG unit versus SISO decoders in our parallel turbo decoder.

Moreover, the area of the parallel interleaved address generation unit grows with a much slower pace than the area of SISO decoders with respect to the degree of parallelism. This result shows that our proposed parallel architecture for UMTS/HSPA+ turbo decoder interleaver is efficiently scalable to highly parallel turbo decoders with high throughputs.

Since our proposed algorithm [3] is the only design with interleaved writing capability (computing interleaved address from original address), no other design with the same functionality exists that we can compare with. However, we compare our design with the most recent implementations of conventional, on-the-fly UMTS interleavers (Table V). Asghar et. al. in [6] and [7] propose a hardware reuse technique to reduce the cost of the parallel UMTS interleaver. To increase the efficiency, the authors propose recursive computation of the intra-row permutation sequence and also recursive computation of indices of this sequence. In addition, they propose doubling the word length of block memory to make

TABLE V. COMPARISON WITH RELATED WORK ON INTERLEAVED ADDRESS GENERATION FOR UMTS/HSPA+ STANDARD. *PL* IS DEGREE OF PARALLELISM.

Impl.	PL	Intlv. Writing	Tech. (nm)	Area (mm^2)	Norm. Area (mm^2)	Freq. (MHz)
[12]	1	No	FPGA–90nm (Cyclone II)	100K gates	0.13	-
[11]	1	No	180	0.24	0.031	130
[8]	1	No	130	0.4	0.102	246
[6]	2	No	65	0.014	0.007	150
[5]	16	No	40	0.12	0.02	350
This work	**32**	**Yes**	**65**	**0.15**	**0.004**	**700**

two simultaneous memory accesses possible. Benkeser et. al. in [8] also propose arithmetic transformations to minimize the computational complexity of interleaving and to optimize the ASIC design of their interleaver, but their design does not support parallelism.

An FPGA implementation of UMTS interleaver is proposed in [12]. Their introduced architecture handles address generation and data streaming, but does not support parallel interleaving. The proposed designs in [5] and [6] support parallelism but none of the previous works support interleaved writing. Therefore, if any of the previous methods are to be used for interleaved writing, they must generate the whole interleaving pattern and store it in memory in advance, which is inefficient.

For a fair area comparison, the normalized area for each implementation is reported; normalization is done with respect to technology node and parallelism. That is, for those implementations which support parallel interleaved address generation (i.e. $PL > 1$) the area is divided by PL. For [12], the area is estimated using the reported gate count and the FPGA technology node of 90nm. The normalized area for our implementation is smaller than all of the other works. The working frequency is also the best among all. Our proposed design supports interleaved writing to memory and therefore by enabling the interleaving scheduling introduced in [3], it offers higher degrees of parallelism and therefore significantly higher throughputs compared to all of the previous works.

VI. CONCLUSIONS

We propose a new algorithm and architecture for an interleaved address generator unit of UMTS/HSPA+ standard turbo decoder. The proposed approach facilitates interleaved writing to turbo decoder memory by generating interleaved addresses from input original addresses on-the-fly and efficiently. Interleaved writing is an essential step in the new turbo decoding memory access scheduling that has been proposed recently to mitigate memory contention in UMTS/HSPA+ parallel interleaver and to improve turbo decoding throughput. In order to generate an interleaved address from an original address, we perform reversed intra-row and inter-row permutations. A key contribution in our method is the use of extended Euclidean algorithm to compute modular multiplicative inverse. We also present a parallel hardware architecture for the introduced interleaved address generator that exploits resource reuse and is efficiently applicable in practical highly parallel turbo decoders. Our design generates 32 interleaved addresses in one cycle and satisfies the data rate requirement of 672 Mbps in HSPA+ while at the same time it offers better silicon area and working frequency in comparison to other recent works.

ACKNOWLEDGMENT

This work was supported in part by Huawei and by the US National Science Foundation under grants CNS-1265332, EECS-1232274, EECS-0925942 and CNS-0923479.

REFERENCES

[1] 3rd Generation Partnership Project (3GPP), "Technical specification Universal Mobile Telecommunications System (UMTS); multiplexing and channel coding (FDD), Tech. Spec. 25.212 Release-11," Sept. 2012.

[2] Nokia Siemens Networks, "Long term HSPA evolution. Mobile broadband evolution beyond 3GPP release 10," 2010.

[3] G. Wang, A. Vosoughi, H. Shen, J. R. Cavallaro, and Y. Guo, "Parallel interleaver architecture with new scheduling scheme for high throughput configurable turbo decoder," International Symposium on Circuits and Systems (ISCAS) 2013, in press.

[4] G. Wang, Y. Sun, J. R. Cavallaro and Y. Guo, "High-throughput contention-free concurrent interleaver architecture for multi-standard turbo decoder," IEEE 22nd International Conference on Application Specific Systems, Architectures and Processors (ASAP), pp. 113-121, 2011.

[5] T. Ilnseher, M. May, and N. Wehn, "A multi-mode 3GPP-TE/HSDPA turbo decoder," in IEEE International Conference on Communication Systems (ICCS), pp. 336 –340, Nov. 2010.

[6] R. Asghar, Wu Di, J. Eilert, and D. Liu, "Memory conflict analysis and interleaver design for parallel turbo decoding supporting HSPA Evolution," 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools (DSD), pp.699-706, Aug. 2009.

[7] R. Asghar and D. Liu, "Towards radix-4, parallel interleaver design to support high-throughput turbo decoding for re-configurability," IEEE Sarnoff Symposium, pp.1-5, Apr. 2010.

[8] F. Speziali and J. Zory, "Scalable and area efficient concurrent interleaver for high throughput turbo-decoders," in Proc. Euromicro Symposium on Digital System Design (DSD), 2004, pp. 334 – 341.

[9] C. Benkeser, A. Burg, T. Cupaiuolo, and Q. Huang, "Design and optimization of an HSDPA turbo decoder ASIC," IEEE Journal of Solid-State Circuits, vol. 44, issue 1, pp. 98-100, Jan. 2009.

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

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

[12] H. Borrayo-Sandoval, R. Parra-Michel, L. Gonzalez-Perez, F. Printzen, and C. Feregrino-Uribe, "Design and implementation of a configurable interleaver/deinterleaver for turbo codes in 3GPP standard," in International Conference on Reconfigurable Computing and FPGAs, pp. 320-325, Dec. 2009.

[13] A. Hassan, M. Shokair, A. A. Elazm, D. Truhachev, and C. Schlegel, "Proposed deterministic interleavers for CCSDS turbo code standard," Journal of Theoretical and Applied Information Technology, vol. 16, no. 1, pp 29-33, Jan. 2010.

[14] J. Hoffstein, J. Pipher, and J.H. Silverman, An Introduction to mathematical cryptography. Springer Publishing Company, Incorporated, 2008

[15] A. Giulietti, B. Bougard, V. Derruder, S. Dupont, J. W. Weijers, and L. V. der Perre, "A 80 Mb/s low-power scalable turbo codec core," Proceedings of Custom Integrated Circuits Conference, pp. 389-392, May 2002.

[16] Y. Sun, "Parallel VLSI architectures for multi-Gbps MIMO communication systems," Ph.D. dissertation, Rice University, Dec. 2010.