An Efficient Hardware Interleaver for 3G Turbo Decoding

Paul Ampadu and Kevin Kornegay
Cornell Broadband Communications Research Lab
pka3@cornell.edu

***Abstract* – We describe an energy-efficient approach for VLSI implementation of the 3rd Generation Partnership Project (3GPP) turbo coding interleaver algorithm. Unlike previous implementations, this interleaver uses a two-stage dedicated hardware datapath that exploits the iterative nature of the decoding process, to compute addresses on the fly, eliminating the overhead associated with programmable processors and precomputed address storage. By separating the interleaving process into two stages, our architecture allows the preparatory phase to be turned off during iterations, while the decoder engages only the real-time address computation phase, further reducing power consumption.**

INTRODUCTION

Turbo codes [1] have been shown to provide superior error performance amidst hostile transmission media. 3rd generation systems based on the 3GPP air interface standard [2] must support turbo coding at data rates of between 384kbps to 2Mbps, while maintaining low mobile station power. The error performance of turbo codes is determined by their distance spectrum, a parameter that is affected both by the encoder structure and the interleaver type and depth. Whereas external interleavers are customarily used to spread out burst errors to a fading channel (e.g. wireless channel), a non-uniform internal interleaver is employed in turbo codes, in order to randomize the data sequence from the encoder and achieve maximum entropy. The interleaver also ensures that the parity sequences generated by the two recursive systematic convolutional encoders are scrambled to appear as uncorrelated as possible, allowing the use of iterative suboptimal decoding algorithms in the decoders. Fig. 1 is a simplified representation of turbo decoding.

A turbo decoder consists of soft-input soft-output (SISO) component decoders, which exchange information cooperatively to produce better estimates of the transmitted symbols over the noisy channel. The estimates, in the form of probabilities, are interleaved and deinterleaved between SISO computations. While the interleavers are not the dominant power or performance consumers of a turbo decoding system, their presence impacts the throughput and memory requirements of the system significantly. By employing efficient algorithm simplification and hardware mapping, we seek to operate the system at the lowest possible frequency, thus reducing overall power consumption.

The 3GPP-defined turbo code interleaver algorithm comprises a series of mathematically complex processes that map an input sequence of length K ($40 \le K \le 5114$) to a scrambled sequence of the same length. The algorithm can be summarized as follows:

1. Row-wise data input to an RxC rectangular matrix, with zero padding if K<RxC.
2. Intra-row permutation of the rectangular matrix, based on a recursively constructed base sequence s.
3. Inter-row permutation of the rectangular matrix, based on a well-defined inter-row permutation pattern T and a permuted least primes sequence r (obtained from T and an ordered least primes sequence q).
4. Column-wise data output from the rectangular matrix, with pruning.

The innovation of our approach is to combine steps 2 through 4 in a single function that allows real-time address computation. We separate the interleaving process into a preparatory phase and a real-time address computation stage. The preparatory phase prepares all the parameters necessary for real-time address computation, using nearly constant time for all sequence lengths. This process determines R, C, p, s, T, and r, as required by the algorithm above. At the end of the preparatory process, a DONE register signals to the real-time address generator that the various operations have been completed, and the needed registers have been prepared. The real-time address generator then uses the prepared sequences to compute addresses on the fly. The two units communicate via simple register control protocols. The decoder processor interacts with the real-time address generation stage through control signals “Interleave” or “Deinterleave”. During decoding iterations, the preparatory phase can be turned off, further saving power. The generalized real-time j^{th} interleaved address is computed using equation (1) below, where the operators have their usual meanings. Fig. 2 is the general architecture of our hardware interleaver.

$$\text{Interleaved\_Address}[j] = C * T[j\%R] + s[((j/R) * r[T[j\%R]])\%(p-1)] \quad (1)$$

The control mechanism for address generation is as follows:

```
Initialize counter
While (counter < K)
  Compute Interleaved_Address [j] //using (1)
  If (Interleaved_Address [j] < K)
          Increment counter
  Increment j
```

Clearly, the latency and computational requirement of the real-time address generation phase are a function of the particular input sequence length.

RESULTS AND DISCUSSIONS

In certain coding experiments where only a few frame lengths are allowed, it is possible to store the entire interleaved sequence for each of the possible frame lengths in a small lookup table. For the 3GPP interleaver, the range of allowable frame lengths ($40 \le K \le 5114$) is quite large, and the scrambled sequences are different for different K. In this case, a brute force approach that stores the interleaved addresses for each of the possible input sequence lengths would require roughly 25MB storage. It is also possible to implement the interleaver using a software programmable processor that prepares an address memory, given a particular value of the input sequence length. This approach would require at least a 5K 13-bit word memory, ignoring the memory required to hold the programming instructions. Our architecture minimizes power and area, by removing storage memory.

We have designed the interleaver using a 0.25μ 6-layer metal CMOS process, and verified its functionality for sequences over the required range. Fig. 3 shows a layout of the real-time interleaver component, whose performance we compare with typical 5K 13-bit ASIC memory storage. While the chip core occupies a $2.54mm^2$ area (a greater percentage of the chip area is attributed to an unoptimized directly implemented divider component) at a comfortable 12.5MHz clock, we expect even greater overall energy savings. We have estimated energy savings at greater than $4\mu J$ for a fixed eight-iteration decoding process in a typical optimized ASIC core in a similar CMOS process. Removal of this memory should translate to an overall reduction of at least $1mm^2$ in interleaver area. In estimating the total energy saving per operation, we have neglected the DC and leakage components usually associated with random access memories, basing our estimates only on the capacitance driven per operation. Our approach, which accepts a slight penalty in computational complexity, further allows portions of the datapath not in use to be turned off, resulting in further reduction in power.

CONCLUSIONS

We have outlined associated challenges in implementing the 3GPP turbo code internal interleaver and described some techniques used to overcome them. In particular, we have described architectural innovations that have allowed us to reduce interleaver latency, power consumption, and area.

We have made our estimates based on worst-case assumptions. While our estimates suggest improved performance gains, it is possible that when the distribution of frame lengths is uniform, the actual performance penalty of the software programmable approach may not be that excessive, compared to our approach. Furthermore, a software programmable approach allows one the flexibility of algorithm improvement and executable updates, a property that is lacking in our approach. Subsequently, actual physical chip measurements will be performed to obtain more accurate performance comparisons.

Acknowledgment

We acknowledge funding and research resources from IBM Research, where initial analysis of this project was undertaken.

References

1. 3rd Generation Partnership Project (3GPP) TSG-RAN: "Multiplexing and Channel Coding," Release 4, Version 4.2.0, Sept. 2001. Document available at www.3gpp.org.
2. C. Berrou, A. Glavieux, and P. Thitimajshima, "Near-Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," Proc. 1993 IEEE Int. Comm. Conf. (Geneva, Switzerland, May 1993), pp. 1064-1070.

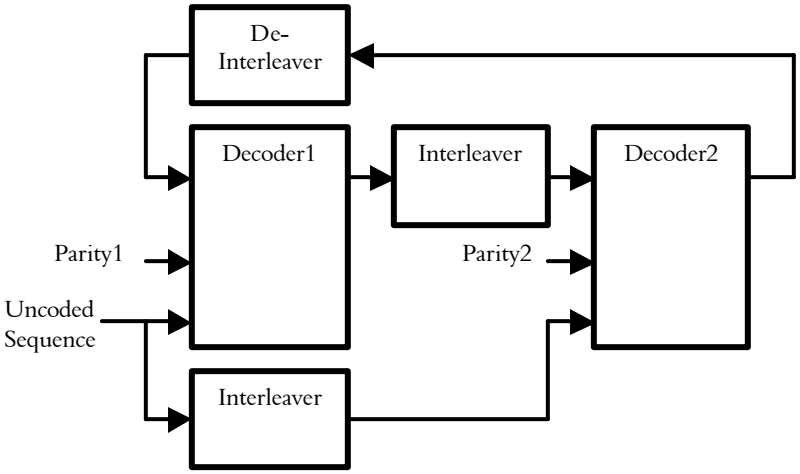

Fig. 1. Turbo decoder architecture

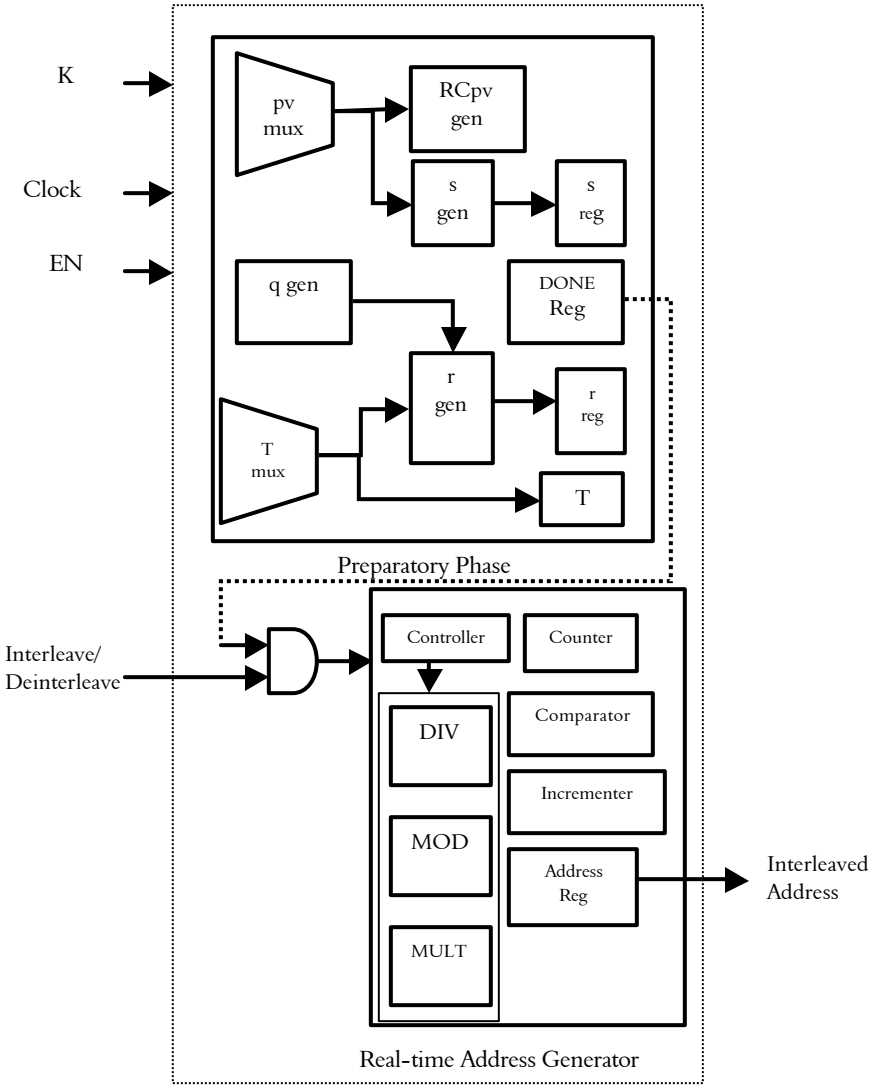

Fig. 2. Two-stage hardware interleaver architecture

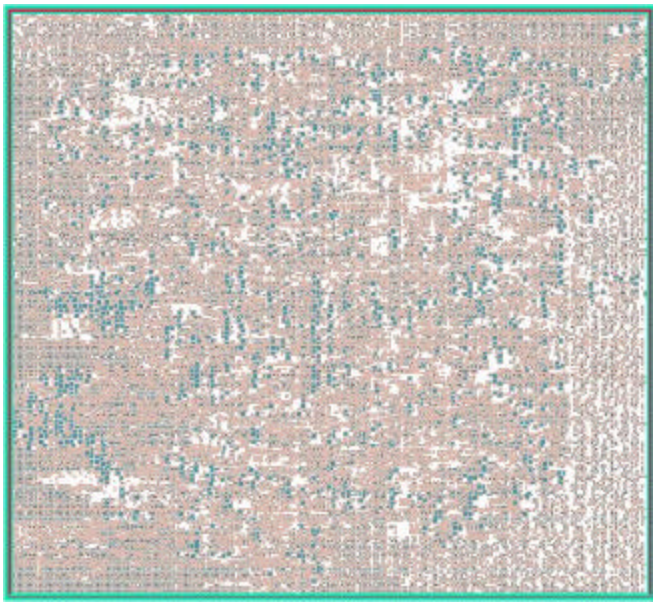

Fig. 3. Interleaver real-time address generator chip layout