CONCURRENT INTERLEAVING ARCHITECTURES FOR HIGH-THROUGHPUT CHANNEL CODING

Michael J. Thul, Frank Gilbert, Norbert Wehn

Microelectronic System Design Research Group, University of Kaiserslautern
Erwin-Schrödinger-Straße, 67663 Kaiserslautern, Germany
{thul, gilbert, wehn}@eit.uni-kl.de

ABSTRACT

Interleavers are widely used for a vast range of communications applications. Traditionally used for burst-error separation in distorted channels, they have gained additional interest since the discovery of Turbo-Codes whose performance essentially depends on the interleavers. With the ever increasing data rates demanded by customers, architectures that provide interleaving at high throughput become mandatory.

We present a heuristic approach to the design of interleaving architectures based on random graph generation. They can handle any given interleaver pattern and allow for any parallelization degree, and hence speed-up, of the interleaving operation.

Moreover, this enables highly parallel architectures for channel decoders such as Turbo- and LDPC-Decoders.

1. INTRODUCTION

Interleaving is scrambling the processing order of the data inside a block to break up neighborhood-relations. It is used in many channel coding schemes and also essential for the communications performance of Turbo-Codes. Interleaver tables contain one-to-one mappings of source addresses to destination addresses for reordering the data. In Table 1 a sample interleaver and deinterleaver is shown, where deinterleaving maps the interleaved data back onto its original sequence. If only one data is interleaved per clock cycle, the reordering can be performed on the fly through indirect addressing.

Address	Interleaved	Address	Deinterleaved
1	3	1	6
2	6	2	4
3	5	3	1
4	2	4	5
5	4	5	3
6	1	6	2

Table 1. Interleaver and Deinterleaver Tables for Six Addresses

source memory	rel. Addr.	*Addr.*	*Interleaved*	target memory	rel. Addr.
⇒ **1**	**1**	***1***	***3***	**1**	**3**
1	2	*2*	*6*	2	3
1	3	*3*	*5*	2	2
⇒ **2**	**1**	***4***	***2***	**1**	**2**
2	2	*5*	*4*	2	1
2	3	*6*	*1*	1	1

Table 2. Interleaver Table with Memory Partitioning

High throughput, however, implies the interleaving of several data per cycle. *Parallel Turbo-Decoders for high throughput, for example, are only enabled through concurrent interleaving architectures [4].* The problem is best illustrated by taking the interleaver table of Table 1 for two concurrently interleaved data and partitioning its addresses and data into two individual memories. Table 2 shows the interleaver table entries together with the associated memories and relative addresses[1].

The number of write accesses can be determined from the interleaver tables and the reading scheme: Assuming that the two data are read in the order of ascending relative addresses (i.e. in the first cycle at the absolute addresses 1 and 4) and interleaving is performed according to Table 2, Table 3 shows the resulting write accesses.

In the first cycle one data is read from source memory 1 (Addr. 1) and written to target memory 1 (Addr. 3). The other one is read concurrently from source memory 2 (Addr. 1) and written to target memory 1 (Addr. 2), which results in two concurrent write accesses for target memory 1.

[1]From now on, only the interleaver is mentioned. Of course, the same concepts apply to the deinterleaver as well.

Clock Cycle	Write Accesses to memory 1	Write Accesses to memory 2
1	2	0
2	0	2
3	1	1

Table 3. Write Accesses to Memories

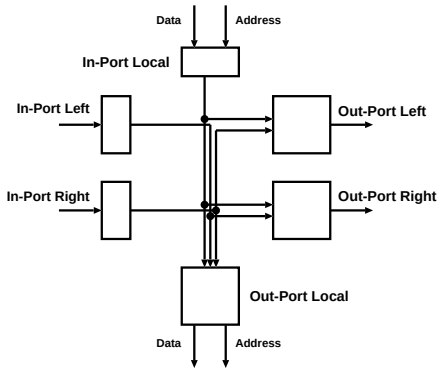

Fig. 1. RIBB Cell

For arbitrary interleavers concurrent write operations to memories occur, thus resulting in conflicts although each memory is *on average* only accessed once per cycle. These conflicts can either be avoided, through the design of a conflict free interleaver as in [1] or by deriving a schedule for optimal read/write sequences for a known interleaver as in [3] or these conflicts can be resolved at run-time. We choose the latter approach as it is the only one providing for architecture-independent and standard-compliant interleavers. Buffers can be introduced to match the worst case to the average case.

2. INTERLEAVING ARCHITECTURES

In [6] we introduced a buffer concept for concurrent interleaving, where special multiple-input FIFOs are added in front of each memory. These FIFOs are capable of storing several data concurrently in one cycle while outputting one data per cycle [6]. Thus the concurrency is moved into those FIFOs.

However, for *parallelization degree* N, the FIFOs scale with $O(N^3)$ in area and pose severe layout problems for larger N. In [5] we presented a new architectural concept, optimized with respect to layout and interconnect properties. A Ring-Interleaver-Bottleneck-Breaker (RIBB) is introduced, which consists of individual cells with one data input and one data output each, plus connections to two neighboring cells, see Figure 1 and Figure 2.

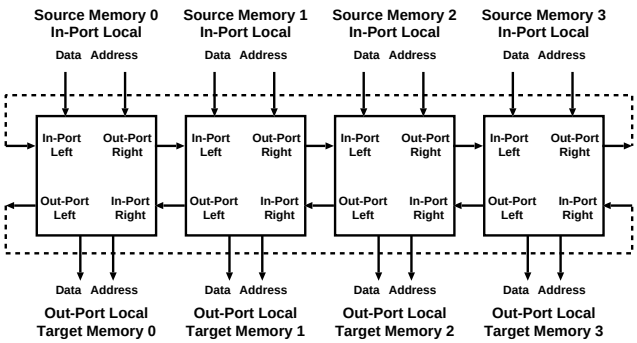

Fig. 2. RIBB

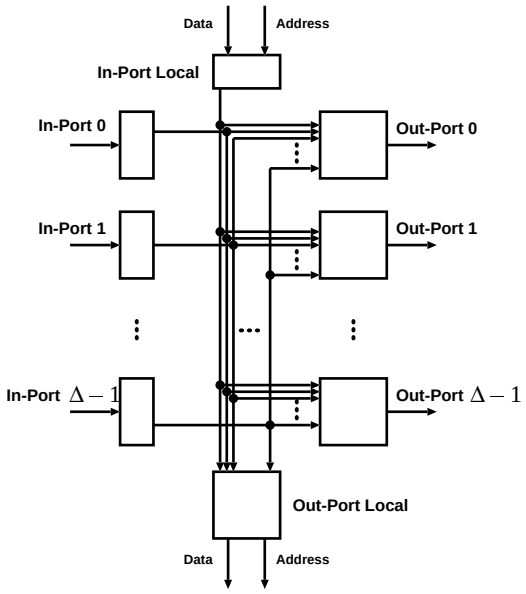

Fig. 3. GIBB Cell

The local in-port determines from the incoming address the target memory and decides whether the data is to be stored locally or to be forwarded as a data/address packet left or right for a shortest paths routing to the target cell. The other in-ports only determine whether the incoming data/address packet is to be stored locally or fed through. The out-ports consist of named multiple-input FIFOs. The cells are connected in a ring to form the RIBB as shown in Figure 2.

When not limiting the connections to only two neighboring nodes we call it a *General-Interleaver-Bottleneck-Breaker* (GIBB). A RIBB is a special case of GIBB with the number of neighbors equal to two. A GIBB-cell also has one data input and one data output, yet features Δ neighbors at the in-ports and Δ neighbors at the out-ports, see Figure 3. The topology of a GIBB, without local data input and output, can depicted by a directed graph with nodes with out-degree Δ. A GIBB cell translates to a node such as shown in Figure 4.

From now on we refer to a *graph* as the *set of nodes N and edges E*, whereas we refer to a *network* as a *graph with associated routing information*, which is the information which path shall lead from one node to another.

Using GIBBs any graph can be used to describe the interconnection topology. Routing can be performed in various

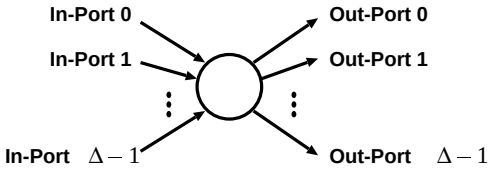

Fig. 4. Node Notation for a GIBB Cell

ways. For the time being we restrict our considerations to a shortest path routing.

3. HIGH-THROUGHPUT REQUIREMENTS

Networks for high-throughput interleaving require a special cost function. We make the following reasonable assumptions:

a) Interleavers spread the data with equal distribution over the address space such that each incoming data has the probability of $1/N$ to be targeted at any specific node k. This holds for interleavers with good communications performance.

b) Each out-port can transmit one data/address packet per clock cycle.

The *traffic* of each out-port is the number of packets to be transmitted over a certain number of cycles. Given assumption a) we can model the traffic during interleaving through the following approach:

Every node communicates with all other nodes. Out of N packets one is kept locally and $N-1$ are transmitted to neighboring nodes. Each out-port is annotated with a counter that keeps track of the passed packets. These counters are incremented as the packets pass the out-port while traveling through the network from node to node until their destination is reached. Though a coverage of all possible interleavers is impossible statistic traffic properties for good interleavers are obtained.

Thus, in N cycles $N \cdot (N-1)$ packets are transmitted. The capacity of the network is sufficient, according to assumption b), when in those N cycles equal or less than N packets pass through each out-port.

Let E be the number of edges, Δ the node out-degree, and d_{jk} the length of the shortest path between the nodes j and k, then the average distance between any two nodes is

$$\bar{D} = \frac{1}{N(N-1)} \cdot \sum_{j,k \in N, j \neq k} d_{j,k}.$$

The $N \cdot (N-1)$ messages must be conveyed in N cycles and each message passes on average $\bar{D}$ edges. Therefore, $(N-1) \cdot \bar{D}$ edges are passed each cycle.

A necessary but not sufficient condition for a network capable of interleaving is: not more than E edges can be passed within one clock cycle, i.e.

$$\begin{aligned} E &\geq \bar{D} \cdot (N-1) \\ \frac{E}{N} &\geq \bar{D} \cdot \frac{N-1}{N} \\ \Delta &\geq \bar{D} \cdot (1 - \frac{1}{N}), \end{aligned} \qquad (1)$$

which also determines the minimum out-degree given the average distance and the number of nodes[2].

[2] Please note that the average distance, in general, depends on the number of nodes.

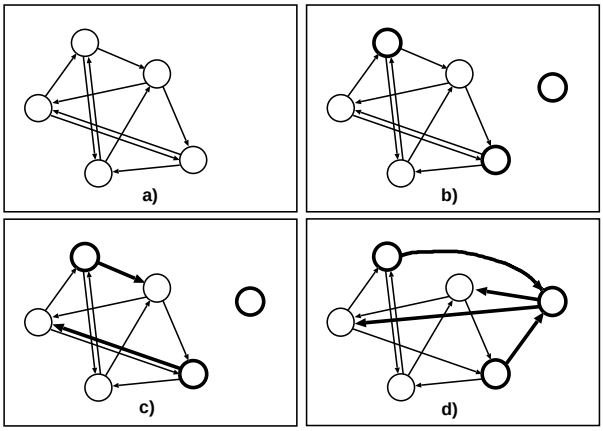

Fig. 5. Random Graph Generation

4. RANDOM GRAPH GENERATION

In [2] random graphs are generated for distributed networks with high fault tolerance, good scalability, and low diameter. The graphs are passed through various filters to check necessary conditions, the accepted ones are evaluated, and the best suited are selected.

In contrast to this, we propose a constructive random graph generation scheme for fixed N and Δ in which graphs are organically grown that are valid by construction. They must fulfill the following conditions: each node has Δ disjunct predecessors and Δ disjunct successors and all nodes can be reached from any given node (connectivity). During growth, the equal distribution property of good interleavers is reflected in the conditions on random selections.

A valid initial graph is given by a fully connected graph of $\Delta + 1$ nodes. It is extended using the following scheme, see also Algorithm 1 and Figure 5: To a given graph of out-degree Δ with M nodes (a) one node is added (b) and connected to Δ randomly selected nodes (c) through randomly selected edges (d) under the condition that only disjunct nodes are connected.

Algorithm 1 Random Graph Generation

{N: Number of Nodes}
{M: Current Number of Nodes}
{Δ: Out-Degree}
a) Create a fully connected graph with $M = \Delta + 1$
while $M < N$ **do**
  b) Add Node
    Randomly $(1/M)$ select Δ disjunct nodes
  c) Randomly $(1/\Delta)$ select one out-ports of each node with disjunct targets
  d) Make targets of the selected out-ports targets of new node
    Make new node the target of selected out-ports
  a) Increment M
end while

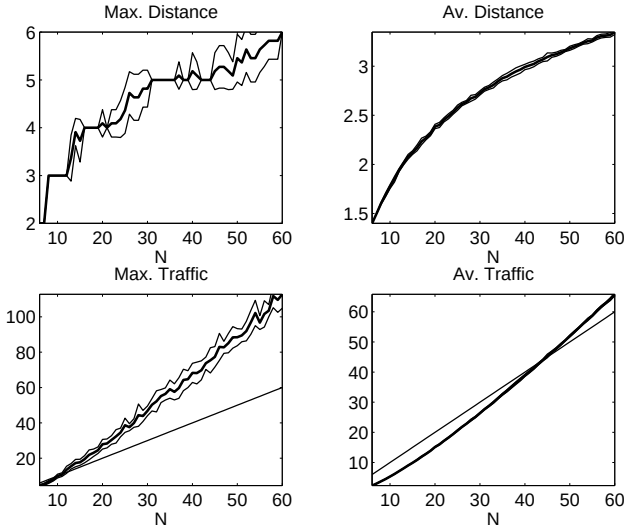

Fig. 6. Statistics of Random Graphs for $\Delta = 3$

5. RESULTS

Figure 6 shows the statistical properties of networks on randomly grown graphs of out-degree $\Delta = 3$. The traffic is derived using a shortest path routing. The average over 1000 graphs is depicted with thick lines, the average +/- standard deviation with slim lines. The average distance of the graphs and average traffic of the networks have a very low variance, whereas the maximum distance varies the most followed by the maximum traffic. The straight lines in the traffic plots depict the bound above which the network capacity is exceeded. Though the maximum traffic exceeds this bound for very low N, the average traffic does not.

Simulated annealing is used for a re-routing that distributes the traffic more evenly onto the edges, thus raising the average, yet decreasing the maximum traffic. The degree of freedom for this optimization is increased through in-ports which feature their own routing table each. Hence data with the same target may be routed through different out-ports of the node depending on the in-port.

Relevant for our exploration is the maximum number of nodes that can be connected using graphs of a given degree. For $\Delta = 3$ we found that up to 43 nodes can be connected with sufficient capacity for interleaving, i.e. the maximum traffic does not exceed 43 after simulated annealing. For $\Delta = 4$ already up to 310 nodes can be connected. The bound given by the average traffic as in Figure 6 would be 44.06. However, the redistribution through simulated annealing must raise the average traffic as it is originally derived from shortest path routing.

The theoretical bound as given in Equation 1 is depicted in Figure 7 with the y-axis normalized to 1. It also suggest the limit of 44.06 nodes. Thus we can conclude that the

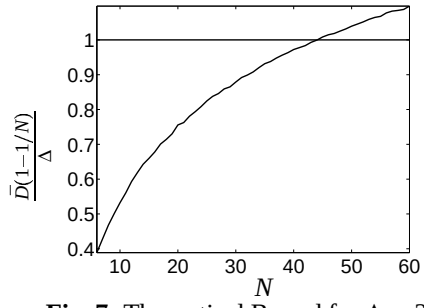

Fig. 7. Theoretical Bound for $\Delta = 3$

average distance is sufficient to estimate the average traffic which is far more complex to determine. A design space exploration can thus focus on the average distance before moving to the maximum traffic which is mandatory for the finishing simulated annealing.

6. CONCLUSION AND FUTURE WORK

The parallelization of concurrent interleaver architectures no longer has a conceptional bound. Given the parallelization, a degree has to be chosen and a graph can be grown to fulfill the capacity requirements of interleaving. *For iterative channel decoders, e.g. Turbo- and LDPC-Decoders, this paves the way for highly parallel architectures for high throughput.*

Further research on General-Interleaver-Bottleneck-Breakers will incorporate physical layout properties into a cost function, intermediate graph selection during growth for improved properties based on evolutionary concepts, and trade-offs between degree and size of individual buffers.

7. REFERENCES

[1] A. Giulietti, L. van der Perre, and M. Strum. Parallel turbo decoding interleavers: avoiding collisions in accesses to storage elements. *Electronics Letters*, 38(5):232–234, Feb. 2002.

[2] V. Lakamraju, I. Koren, and C. Krishna. Synthesis of interconnection networks: A novel approach. In *in Proc. 20th International Conference on Dependable Systems and Networks*, pages 56–64, June 2000.

[3] T. Richter and G. Fettweis. Parallel Interleaving on Parallel DSP Architectures. *Proc. 2002 Workshop on Signal Processing Systems (SiPS '02)*, pages 195–200, Oct. 2002.

[4] M. J. Thul, F. Gilbert, T. Vogt, G. Kreiselmaier, and N. Wehn. A Scalable System Architecture for High-Throughput Turbo-Decoders. In *Proc. 2002 Workshop on Signal Processing Systems (SiPS 2002)*, San Diego, California, USA, Oct. 2002.

[5] M. J. Thul, F. Gilbert, and N. Wehn. Optimized Concurrent Interleaving for High-Speed Turbo-Decoding. In *Proc. 9th IEEE International Conference on Electronics, Circuits and Systems - ICECS 2002*, Dubrovnik, Croatia, Sept. 2002.

[6] M. J. Thul, N. Wehn, and L. P. Rao. Enabling High-Speed Turbo-Decoding Through Concurrent Interleaving. In *Proc. 2002 IEEE International Symposium on Circuits and Systems (ISCAS '02)*, Phoenix, Arizona, USA, May 2002.