Low-Latency SISO via Message Passing on a Binary Tree *

Phunsak Thiennviboon and Keith M. Chugg
Dept. of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089-2565
{thiennvi,chugg}@usc.edu

Calculating the “soft inverse” of a deterministic finite-state machine (FSM) via the SISO (soft-in/soft-out) module is a key operation in many data detection/decoding algorithms [1, 3]. Perhaps the most appreciated application is iterative decoding of concatenated codes. The standard SISO algorithm is the forward-backward algorithm (e.g., [1, 3]) which can be derived from the generalized distributive law (GDL) algorithm [2] in a linear-structure junction tree as shown in Fig. 1(a). Note that $a_k$, $s_k$, $t_k$, and $x_k$ are input, state, transition, and output at time index $k$ of the FSM where $x_k = \mathcal{P}_k(s_k, a_k)$, $s_{k+1} = \mathcal{Q}_k(s_k, a_k)$, and $t_k = (a_k, s_k, s_{k+1}, x_k)$. The components in the dashed block define the SISO module. The local kernels of the nodes $a_k$ and $x_k$ are the soft-input for each variable and the local kernel of the node $t_k$ is a local validity check for all variables of the node. The message passing schedule for this junction tree can be assigned by 3 familiar steps, i.e., forward recursion, backward recursion, and combining/completion. Note that the associated forward and backward recursion steps can be computed in parallel for all of the FSM states at a given time. This architecture provides $\mathcal{O}(N)$ latency and computational complexity, where $N$ is the block size.

![Figure 1: Junction tree of SISO algorithm with (a) linear (b) binary-tree structure](image)

In order to achieve a lower latency algorithm, we arrange the structure of the junction tree to be a binary tree by introducing additional local domains, $\{v_{i,j}\}$, as shown in Fig. 1(b) for $N = 8$. The local kernel of node $v_{i,j}$ is a local validity check for all variables of the node. Since both junction trees in Fig. 1 represent the same system, message passing (i.e., the GDL) on either produces the same desired soft inverse. Considering the

---

*This work was supported in part by the National Science Foundation (NCR-CCR-9726391)*
activating schedule, all nodes at the same depth of the binary tree are activated simultaneously from the leaf nodes \( a_k \) and \( x_k \) to the root node \( v_{0,0} \) (or inward recursion), and then the process continues in the reverse direction (or outward recursion). Note that, the pending messages from nodes \( a_k \) and \( x_k \) to \( t_k \) are the soft-input, SI[\( a_k \)] and SI[\( x_k \)], and, after finishing outward recursion, the messages from node \( t_k \) to nodes \( a_k \) and \( x_k \) are the soft-output, \( \text{SO}[a_k] \) and \( \text{SO}[x_k] \) (see [3] for detailed description of the messages). We refer to this SISO as the **Forward-Backward-Tree SISO (FBT-SISO)**. Using this inward/outward schedule, it is straightforward to show that the latency of this algorithm is \( 2 \log_2 N \). Since the message updating on a node associated with 3 states typically has a highest complexity, the computational complexity of this algorithm is \( \mathcal{O}(S^9 N) \) where \( S \) is maximum cardinality of all states in the FSM. Therefore, the FBT-SISO has \( \mathcal{O}(\log_2 N) \) latency and \( \mathcal{O}(N) \) computational complexity. Note that this binary-tree structure can be applied easily for an arbitrary value of \( N \). As an example, let consider \( N = 2^m \). The additional local domain \( v_{i,j} \) \((i \in \{0, 1, \ldots, m - 1\}, j \in \{0, 1, \ldots, 2^i - 1\}\) can be defined as follows: \( v_{0,0} = s_{a[0]} \); \( v_{i \neq 0,0} = (s_{a[i]}, s_{x[i]}) \); \( v_{i \neq 0,j \neq \{0,2^i-1\}} = (s_{\{2j\}a[i]}, s_{\{2j+1\}a[i]}, s_{\{2j+2\}a[i]}) \); and \( v_{i \neq 0,2^i-1} = (s_{\{2^{i+1}-2\}a[i]}, s_{\{2^{i+1}-1\}a[i]}) \) where \( \alpha(i) = 2^m-1 \), \( i \) is the depth of node \( v_{i,j} \) \((i = 0 \text{ for root node } v_{0,0}) \), \( j \) is the position (from left to right) of node \( v_{i,j} \) at depth \( i \).

Recently, it has been demonstrated that the SISO computations can be done using a combination of prefix and suffix operations, which leads to a tree architecture with \( \mathcal{O}(\log_2 N) \) latency [4]. This architecture is based on well-known tree-structures for fast parallel prefix computations in the Very Large Scale Integration (VLSI) literature (e.g., fast adders [5, 6, 7]), so we refer to it as a **tree-SISO**. As compared to the tree-SISO, the FBT-SISO presented herein has double the latency but is less (computationally) complex by a factor of approximately \( \log_2 N \). Finally, there exist algorithms with a parallel prefix computation which have similar structure to the FBT-SISO, e.g., the fast adder in [7].

### References


