Performance Analysis of Latency-Insensitive Systems

Ruibing Lu and Cheng-Kok Koh, *Member, IEEE*

***Abstract*—This paper formally models and studies latency-insensitive systems (LISs) through max-plus algebra. We introduce state traces to model behaviors of LISs and obtain a formally proved performance upper bound achievable by latency-insensitive design. An implementation of the latency-insensitive protocol that can provide robust communication through back-pressure is also proposed. The intrinsic performance of the proposed implementation is acquired based on state traces. It is also proved that the proposed implementation can always reach the best performance achievable by latency-insensitive design.**

***Index Terms*—Back-pressure, latency-insensitive system, max-plus algebra, performance analysis, state trace.**

I. Introduction

As the system complexity increases and the feature size scales down to deep submicrometer dimensions, the interconnect delay becomes a dominating factor of the system performance. Although the delay of interconnects can be significantly reduced by such interconnect optimization techniques [1] as topology optimization, wire sizing, and buffer insertion and sizing, the gap between delays of interconnects and devices continues to grow. With increasing operating frequencies and chip sizes of integrated circuits (ICs), the delay from one circuit block to another may be longer than one clock period. In fact, the delay of global interconnects can be as long as about five to ten clock cycles in the near future [2], making pipelined global communication inevitable. Unfortunately, the estimations of the delay and latency of global interconnects at high-level design stages have very low accuracy. Moreover, the insertion of flip-flops [3]–[5] and the change of interconnect latency without considering global consistency affect circuit functionality and destroy the correctness of system behavior. Retiming [6] with pipelined interconnects [7]–[9] has been proposed to move flip-flops to the middle of interconnects so as to remove timing violations while maintaining the functionality of circuits. The effectiveness of interconnect retiming however, is quite limited as flip-flops available for redistribution are typically limited. Architectural retiming [10], [11] is proposed to increase the number of registers on critical paths by using negative registers, which are required to precompute or predict signal values. Such negative registers may be difficult to implement for general global interconnects.

A latency-insensitive-design methodology, which can tolerate the change of communication latencies in late design stages of synchronous systems, has been proposed in [12]. Carloni *et al.* [13] have presented an implementation of such latency-insensitive systems (LISs), in which the timing violations of communication channels can be removed by pipelining the channels with relay stations. The performance of LISs has been analyzed in [14]. This analysis however, ignores the "back-pressure," which is necessary to establish fully reliable communications by informing the source module to stall when the sink module cannot accommodate more input data [13], [15]. Clearly, LISs without back-pressure is much less robust than those with back-pressure.

In this paper, the behaviors of LISs are formally modeled and studied based on max-plus algebra [16]. We also propose an implementation of the latency-insensitive protocol that not only provides robust communications through back-pressure, but can always reach the best performance achievable by latency-insensitive design.

II. Background and Related Research

Latency-insensitive-design methodology shares with asynchronous-circuit design [17] some common ideas and characteristics, e.g., "handshaking signaling" of communications. However, the most important difference is that latency-insensitive design is synchronous and can be implemented using the conventional design flow of synchronous circuits with minor changes.

An LIS is composed of a collection of modules that exchange data using communication links. The communication among modules of an LIS is controlled by the latency-insensitive protocol. The precondition of latency-insensitive design is that any module can freeze its operation, or "stall," for an arbitrary time without losing their internal states. A module has to stall unless each of its input channels can provide informative data; that provides a method for synchronization of input data when input channels have different latencies. When a module stalls, it produces noninformative data to its output channels. Informative data already generated but not ready to be consumed are stored in channel queues temporarily. Therefore, the protocol guarantees the correct behaviors of an LIS independent of communication latencies. The change of communication latency is performed by inserting relay stations, which can be used to pipeline the communication like registers but with handshaking signaling. A module may also stall when any

Manuscript received March 17, 2004; revised July 26, 2004 and December 20, 2004. This work was supported in part by the National Science Foundation under Grant CCR-9984553. This paper was recommended by Associate Editor S. Hassoun.

R. Lu is with Synopsys Inc., Mountain View, CA 94043 USA (e-mail: Ruibing.Lu@synopsys.com).

C.-K. Koh is with School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: chengkok@purdue.edu).

Digital Object Identifier 10.1109/TCAD.2005.854636

of its output channels cannot accommodate more informative data; this is called "back-pressure."

The theory of latency-insensitive design is introduced in [12]. Carloni *et al.* have presented an implementation of such LISs in [13]. In order to fulfill latency-insensitive communication protocol, two additional interconnects with opposite directions are added for each communication channel; the one that has the same direction as data transmission is used to identify whether the data are informative and the other one is to provide back-pressure. The performance of LISs without back-pressure is studied in [14]. However, to the best of our knowledge, there is no published performance-analysis method for LISs described in [13].

The problem of determining the system performance has been studied in several other contexts. Rate analysis of embedded systems has been performed through graph-based techniques [18]. In [19] and [20], the worst case timing analysis has been performed for embedded systems, while the average case performance has been studied in [21]–[23]. The performance of asynchronous systems has also been extensively studied. Analytical methods [24]–[26] to obtain exact measures of performance have been proposed for a system with deterministic delays. Systems with general delay distributions have been studied with stochastic-based techniques in [27]–[29] In [16], discrete-event dynamic systems are studied with algebraic techniques and theory of linear systems, which we adopt in this work as the theoretical basis to study the behavior of latency-insensitive design.

In the remainder of this section, we first introduce the basic notation and definitions for max-plus algebra that will be used to model LISs. A brief overview of previous studies, especially the performance analysis, for latency-insensitive design are then outlined and discussed.

A. Max-Plus Algebra

In this paper, we denote the set of the real numbers by $\mathbb{R}$, the set of nonnegative integers by $\mathbb{N}$, and the set of positive integers by $\mathbb{N}_0$. The number of elements of a set S is denoted by $|S|$. Let $A \in \mathbb{R}^{m\times n}$ be an $m \times n$ matrix; the (i,j)th entry is denoted by $a_{i,j}$ or $(A)_{i,j}$. $X \in \mathbb{R}^n$ be a vector with n entries. The ith entry is denoted by x_i or $(X)_i$. Given two vectors $X, Y \in \mathbb{R}^n$, $X \leqslant Y$ means that $x_i \leqslant y_i$, $\forall i \in \{1,2,\ldots,n\}$. Max-plus algebra is a widely used modeling framework for discrete-event systems [16]. The algebra is based on two operations, which are formally named addition and multiplication and, respectively, denoted by $\oplus$ and $\otimes$, defined as follows[1]:

$$\text{Addition}: \quad a \oplus b \stackrel{\text{def}}{=} \min\{a,b\} \tag{1}$$

$$\text{Multiplication}: \quad a \otimes b \stackrel{\text{def}}{=} a + b \tag{2}$$

[1]The traditional term "max-plus algebra" refers to the semiring $\mathbb{R}_{\max} \stackrel{\text{def}}{=} (\mathbb{R} \cup \{-\infty\}, \max, +)$, that is to the set $(\mathbb{R} \cup \{-\infty\})$ equipped with max as addition (denoted by $\oplus$) and $+$ as multiplication (denoted by $\otimes$). The algebraic structure can be equally applied to $(\mathbb{R} \cup \{\infty\}, \min, +)$, which is sometimes also called as "min-plus algebra." Both of them are special cases of a more general algebra called "dioid" [16]. For convenience, the max-plus algebra in this paper refers to the semiring $(\mathbb{R} \cup \{\infty\}, \min, +)$.

where $a, b \in \mathbb{R}_\infty$ and $\mathbb{R}_\infty \stackrel{\text{def}}{=} \mathbb{R} \cup \{\infty\}$. Similar to regular algebra, commutativity of $\oplus$ and $\otimes$, associativity of $\oplus$ and $\otimes$, and distributivity of $\otimes$ over $\oplus$ hold for the max-plus algebra.

The operations $\oplus$ and $\otimes$ can be extended to matrices. If $d \in \mathbb{R}_\infty$, $A, B \in \mathbb{R}_\infty^{m\times n}$, and $C \in \mathbb{R}_\infty^{n\times p}$

$$(A \oplus B)_{i,j} \stackrel{\text{def}}{=} a_{i,j} \oplus b_{i,j}$$

$$(d \otimes A)_{i,j} \stackrel{\text{def}}{=} d \otimes a_{i,j}$$

$$(A \otimes C)_{i,j} \stackrel{\text{def}}{=} \bigoplus_{k=1}^{n} a_{i,k} \otimes c_{k,j}.$$

The kth ($k \in \mathbb{N}$) max-plus-algebraic power of matrix A is denoted by $A^{\otimes^k}$ and defined recursively as follows:

$$A^{\otimes^1} \stackrel{\text{def}}{=} A$$

$$A^{\otimes^k} \stackrel{\text{def}}{=} A \otimes A^{\otimes^{k-1}}, \qquad \text{for } k = 2,3,4,\ldots.$$

Example 2.1: The following is an example of matrix multiplication in max-plus algebra:

$$\begin{aligned}\begin{pmatrix} 1 & \infty \\ 2 & 1 \end{pmatrix} \otimes \begin{pmatrix} 2 & 1 \\ 0 & \infty \end{pmatrix} &= \begin{pmatrix} \min(1+2, \infty+0) & \min(1+1, \infty+\infty) \\ \min(2+2, 1+0) & \min(2+1, 1+\infty) \end{pmatrix} \\ &= \begin{pmatrix} 3 & 2 \\ 1 & 3 \end{pmatrix}.\end{aligned}$$

B. Precedence Graph

A matrix in max-plus algebra can be represented by a corresponding graph. Given a matrix $A \in \mathbb{R}_\infty^{n\times n}$, we can construct a directed weighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, w)$, called the precedence graph of matrix A, as follows:

$$\mathcal{V} = \{1,2,3,\ldots,n\}$$

$$\mathcal{E} = \{(i,j) | (A)_{j,i} \neq \infty\}$$

$$w(i,j) = (A)_{j,i} \qquad \forall (i,j) \in \mathcal{E}$$

where $\mathcal{V}$ is the vertex set, $\mathcal{E}$ the edge set, and $w : E \to \mathbb{R}$ the edge-weight function. The precedence graph of a given matrix $A \in \mathbb{R}_\infty^{n\times n}$ is denoted by $\mathcal{G}(A)$.

A directed graph is weakly connected, or simply connected, if there is an undirected path between any pair of vertices. A directed graph is strongly connected if there is a directed path between every pair of vertices.

Consider a directed weighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, w)$, the mean weight of a cycle C is defined as the sum of the weights of the edges of this cycle, divided by the length of this cycle. The mean weight of a cycle is also called the cycle mean,

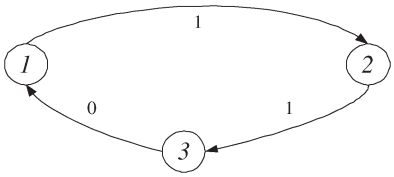

Fig. 1. Example strongly connected precedence graph $\mathcal{G}(A)$. The graph just includes one cycle with mean weight of 2/3, therefore, $\lambda^*(\mathcal{G}(A)) = 2/3$.

and denoted by $\lambda(C)$. The minimum cycle mean of graph $\mathcal{G}$ is defined as

$$\lambda^*(\mathcal{G}) \stackrel{\text{def}}{=} \min_{C\in\mathcal{G}} \lambda(C)$$

which is the minimum mean weight of all cycles in graph $\mathcal{G}$. A cycle whose mean weight is equal to $\lambda^*(\mathcal{G})$ is called a critical cycle. The problem of determining the minimum cycle mean is well studied, and there are several efficient algorithms [30], [31] for solving it.

The following result can be found in many studies, including [32]–[35].

Theorem 2.1: Given $A \in \mathbb{R}_{\infty}^{n\times n}$, if $\mathcal{G}(A)$ is strongly connected, then

$$\exists k_0, d \in \mathbb{N}, \text{ such that } \forall k \geqslant k_0, A^{\otimes^{k+d}} = (\lambda^*)^{\otimes^d} \otimes A^{\otimes^k}$$

where $\lambda^* = \lambda^*(\mathcal{G}(A))$ is the minimum cycle mean of $\mathcal{G}(A)$.

The equation in Theorem 2.1 can be rewritten with traditional algebra as

$$\forall i,j \in \{1,2,\ldots,n\}, \left(A^{\otimes^{k+d}}\right)_{i,j} = (\lambda^* \times d) + \left(A^{\otimes^k}\right)_{i,j}.$$

It means that the power series $\{A^{\otimes^1}, A^{\otimes^2}, A^{\otimes^3}, \ldots\}$ is cyclic if $\mathcal{G}(A)$ is strongly connected.

Example 2.2: Consider an example matrix $A \in \mathbb{R}_{\infty}^{3\times 3}$

$$A = \begin{pmatrix} \infty & \infty & 0 \\ 1 & \infty & \infty \\ \infty & 1 & \infty \end{pmatrix}.$$

The power series $\{A^{\otimes^1}, A^{\otimes^2}, A^{\otimes^3}, A^{\otimes^4}, \ldots\}$ is

$$\left\{ \begin{pmatrix} \infty & \infty & 0 \\ 1 & \infty & \infty \\ \infty & 1 & \infty \end{pmatrix}, \begin{pmatrix} \infty & 1 & \infty \\ \infty & \infty & 1 \\ 2 & \infty & \infty \end{pmatrix}, \begin{pmatrix} 2 & \infty & \infty \\ \infty & 2 & \infty \\ \infty & \infty & 2 \end{pmatrix}, \begin{pmatrix} \infty & \infty & 2 \\ 3 & \infty & \infty \\ \infty & 3 & \infty \end{pmatrix}, \ldots \right\}.$$

It can be observed that $A^{\otimes^4} = 2 \otimes A^{\otimes^1}$. Based on this, we have

$$A^{\otimes^{k+3}} = A^{\otimes^4} \otimes A^{\otimes^{k-1}} = 2 \otimes A^{\otimes^1} \otimes A^{\otimes^{k-1}} = 2 \otimes A^{\otimes^k}, \qquad \forall k > 1.$$

Compare this with Theorem 2.1, we have the period $d = 3$ and $\lambda^* \times d = 2$. Then, $\lambda^* = 2/3$. The precedence graph $\mathcal{G}(A)$ is shown in Fig. 1, and its minimum cycle mean is exactly 2/3.

Theorem 2.1 has important applications for analysis of discrete systems. When a discrete system can be modeled with the power series of a matrix, it can help to find how the system state changes over time. In addition, the cyclicity of the power series can also be used to estimate the speed and period of state changes, which are typically closely related to system performance.

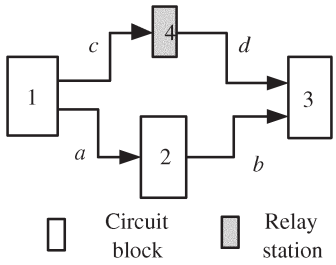

Fig. 2. Simple LIS with three circuit blocks and one relay station.

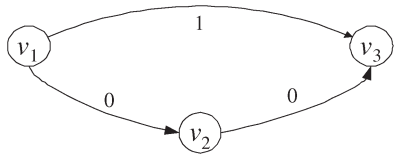

Fig. 3. LIS graph for the LIS shown in Fig. 2. Weight of edge (v_1, v_3) is 1 because there is a relay station between two circuit blocks.

C. Latency-Insensitive Systems

Here, we review the analysis presented in [14]. To model the structure of LISs, Carloni *et al.* [14] introduce the concept of a LIS graph, which is defined as follows

Definition 2.1: A LIS graph $G_{\mathrm{L}}(V, E, w)$ is a weighted connected directed graph, where V is the set of all circuit blocks, $(v_i, v_j) \in E$ refers to the communication link from circuit block v_i to circuit block v_j, and edge weight $w(v_i, v_j) \in \mathbb{N}_0$ is the number of relay stations on the communication link between circuit blocks v_i and v_j.

Example 2.3: Fig. 2 provides an example LIS with three circuit blocks and one relay station, which is on the channel between circuit blocks 1 and 3. Fig. 3 is the corresponding LIS graph, which has one vertex for each circuit block. The weight of edge (v_1, v_3) is 1 because of relay station 4 in the LIS.

For LISs studied in [14], it is assumed that there is no back-pressure for every channel. The performance of an LIS is evaluated based on the throughput in [14]. The throughput of a module can be viewed as the number of informative data produced in one timestamp by this module on average. It is shown in [14] that the throughput of an LIS G_{L}, if G_{L} has cycles, is

$$\vartheta(G_{\mathrm{L}}) = \min_{C\in G_{\mathrm{L}}} \frac{|C|}{w(C) + |C|}. \tag{3}$$

For a cycle C, $w(C)$ is actually the number of relay stations in the cycle, and $|C|$ is the number of circuit blocks. When the system does not have any cycles, the throughput is 1. For an LIS with more than one strongly connected component, infinite queues may be required for communication channels between components of different throughput rates. As infinite queues are impossible in practice, "equalization" is performed in [14] to maintain correct system behavior. The equalization step

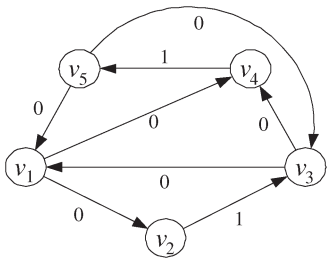

Fig. 4. Example for equalization. This is a strongly connected component S_1 in a LIS graph; the component includes two relay stations on edges (v_2, v_3) and (v_4, v_5), respectively. The critical cycle is $C_a = \{v_1, v_2, v_3, v_4, v_5, v_1\}$ and the component throughput $\vartheta(S_1) = 5/7$. The equalization requires inserting one additional relay station to the critical cycle C_a and expect that the component throughput becomes 5/8. However, one cycle among $C_b = \{v_1, v_2, v_3, v_1\}$, $C_c = \{v_1, v_4, v_5, v_1\}$, and $C_d = \{v_3, v_4, v_5, v_3\}$ becomes the new critical cycle of the component and the new throughput actually becomes 3/5, no matter where the additional relay station is placed.

basically equalizes the throughput rates of all components by adding additional relay stations. For an LIS G_{L} with m strongly connected components, the equalization method proposed in [14] is performed as follows.

1) Compute the throughput $\vartheta(S_k) = a_k/b_k$ for each strongly connected component $S_k \in G_{\mathrm{L}}$.
2) Find $n_k \in \mathbb{N}$ for each $S_k \in G_{\mathrm{L}}$, such that $a_1/(b_1 + n_1) = a_2/(b_2 + n_2) = \cdots = a_m/(b_m + n_m)$.
3) Distribute n_k additional relay stations to the critical cycle C_k of each S_k, and obtain a new LIS G'_{L}.

This method has some limitations. First, usually we have $\vartheta(G'_{\mathrm{L}}) < \vartheta(G_{\mathrm{L}})$ and the performance loss due to equalization, i.e., $(\vartheta(G_{\mathrm{L}}) - \vartheta(G'_{\mathrm{L}}))$, may be significant even if there are a very few number of strongly connected components in G_{L} (see Example 2.4). Second, after the distribution of additional relay stations to the critical cycle of a strongly connected component, it is possible that the component now has a new critical cycle, making the actual throughput less than the expected value. As a result, different components still have different throughputs (see Example 2.5). While other equalization algorithms may solve this problem, it is not clear how the system performance is affected.

Example 2.4: Consider an LIS G_{L} with two strongly connected components to be equalized: $\vartheta(S_1) = 4/5$ and $\vartheta(S_2) = 3/4$. The initial throughput is $\vartheta(G_{\mathrm{L}}) = 3/4$ if the channel queues have infinite capacity. The equalization result is: $n_1 = 3$, and $n_2 = 2$. The throughput after equalization becomes $\vartheta(G'_{\mathrm{L}}) = 1/2$. The performance decreases by 33.3% compared with the initial throughput.

Example 2.5: LIS G_{L} has two strongly connected components to be equalized: $\vartheta(S_1) = 5/7$, and $\vartheta(S_2) = 10/15$. S_1 is shown in Fig. 4. The equalization result is $n_1 = 1$, and $n_2 = 1$, with the expected throughput of 5/8. However, after the addition of one relay station to the critical cycle of S_1, its throughput becomes 3/5, instead of the expected throughput of 5/8. The reason is that the insertion of relay stations may introduce new critical cycles.

These limitations are not inherent in the latency-insensitive-design methodology. In fact, the issue of equalization would not arise at all for an LIS with back-pressure because all modules in such an LIS, as we shall see later, always have the same throughput.

While latency-insensitive design may be used for the assembly of complex Internet Protocol (IP) cores, in this paper, we focus on synchronous LISs with a global clock signal. We assume that the circuit blocks have the following properties.

1) Any path between a block input port and a block output port has at least one register. In other words, there is no combinational path between input and output ports.
2) In each clock cycle, a block takes one set of data from input channels and produces one set of data to output channels.

The first property ensures the screening of noninformative data and prevents them from passing through circuit blocks. The second property simplifies the modeling and analysis of LISs. In practical systems, a block may not have output data in each clock cycle. A simple but not efficient approach is to use dummy output data, which are regarded as informative during modeling and analysis. More realistic and accurate analysis would require the consideration of the way how each circuit block processes input/output data, which is not considered in this paper.

III. LISs With Back-Pressure

In this section, we introduce an implementation of the latency-insensitive protocol with back-pressure. Modules in an LIS are classified into two types: circuit blocks and relay stations. While circuit blocks are required by the system itself to perform specific functions, relay stations are inserted later to resolve timing violations due to a long interconnect delay.

As LISs in this study are modeled differently from those in [13] and [14], we prefix all system elements in [13] and [14] with "B-" and all system elements in the proposed method with "P-." A relay station in [13] and [14] for example, is called a "B-relay station," whereas a "P-relay station" is a relay station in the proposed modeling method. A B-channel in [13] and [14] refers to the communication link between two B-circuit blocks including both wires and B-relay stations inserted into the communication link. In this paper, a P-channel is between two consecutive P-modules, each of which can be either a P-relay station or a P-circuit block. For example, if one P-relay station is inserted into a communication link between two P-circuit blocks, two P-channels are formed from the viewpoint of this paper. In LISs, a module may stall at some timestamps, therefore, proper buffering is required for informative data already generated but not ready to be consumed. This buffering of informative data is modeled as queues in P-channels. In [13], encapsulation shells for B-circuit blocks are used to synchronize input data (arriving with different latencies) and propagate appropriate output values (informative or noninformative) to the output channels. An encapsulation shell includes queues to buffer informative data not ready to be consumed. Fig. 5 shows the two different structures [in Fig. 5(a) and (b), respectively], as well as the mapping between them. A B-relay station is composed of two registers: a main register (MR) and an auxiliary register (AR). The main register can be mapped to the register in a P-relay station, while the AR can be viewed as a queue of

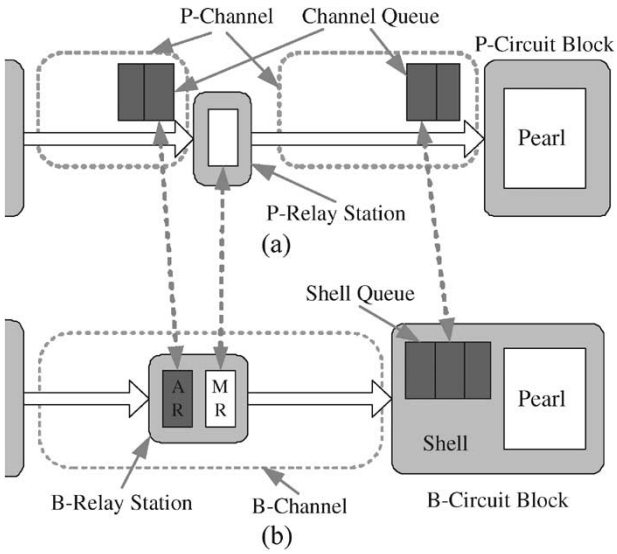

Fig. 5. Comparison of the system structure in [13] and [14] and that in this paper. The AR in a B-relay station can be viewed as the input P-channel queue of size 1 of the corresponding B-relay station. The shell queues of a B-circuit block can be mapped to the input P-channel queues of the corresponding P-circuit block.

size 1 in the input P-channel of the P-relay station. The queues in the shell of a B-circuit block corresponds to the queues of input P-channel of the P-circuit block.

In the proposed implementation of latency-insensitive design with back-pressure, there is one channel queue for each P-channel. Compared with LISs with back-pressure in [13], our implementation has uniform rules for the operation (to compute or to stall) of both P-circuit blocks and P-relay stations, which are completely determined by channel states using a finite state machine (FSM). The uniform operation rules for both P-circuit blocks and P-relay stations enable our performance-analysis method to obtain the exact throughput performance of systems without external influences due to primary inputs and outputs. In addition, a P-relay station can have multiple sink P-modules and support a multiterminal net, which may reduce the number of P-relay stations in LISs. The flexibility of input channel-queue sizes of P-relay stations may also help to reduce the total number of buffering registers. When a P-relay station has multiple sink modules, the same buffering ability may be kept by up sizing the input channel queue of the relay station and down sizing all its output P-channel queues. As the focus of the remainder of this paper is on the performance analysis of LISs based on the proposed implementation, we omit the prefixes for P-channels, P-relay stations, and P-circuit blocks for simplicity.

Now, we present our implementation of the latency-insensitive protocol. Every communication channel has one queue with size equal to or larger than 1, regardless of whether the sink module is a circuit block or a relay station. Otherwise, the informative output of the source module might be lost as follows: When the sink module stalls and cannot accept the informative data of the source module at timestamp t, the source module may receive the stall request (SR) only at timestamp $t+1$. However, the source module may already discard the informative data of timestamp t and generate a new informative data, resulting in the loss of the informative output of the source module at timestamp t. A channel queue with size of 1 is called "minimum queue."

Because the buffering of informative data is performed by the input channel queue, a relay station becomes a register whose output is noninformative in the first timestamp. All circuit blocks generate informative outputs in the first timestamp. Other than that, both circuit blocks and relay stations work in a uniform way controlled by the states of input and output channels. When a channel queue is full and the output of the source module is informative at timestamp t, if the sink module stalls and the source module does not at timestamp $t+1$, a new informative data will be pushed into the already full channel, resulting in channel overflow. Therefore, a channel should request its source module to stall at timestamp t in such a case. A channel is said to be full if and only if the channel queue is full and the output of the source module is informative, and a channel is said to be empty if and only if the channel queue is empty and the output of the source module is noninformative.

For the proposed LISs with back-pressure, there are two conditions under which a module should stall.

1) At least one of the module's input channels is empty, i.e., it cannot provide required informative data to the module.
2) At least one of the module's output channels is full, i.e., more informative data may result in channel overflow.

The behavior of such systems can be described by FSMs. There are three possible states for a channel: informative event (IE), empty data (ED), or SR. A channel is in the ED state if the queue is empty and the output of the source module is noninformative. A channel is in the SR state if the queue is full and the output of the source module is informative. In all other situations, the channel is in the IE state. For a channel with queue size q, the IE state is further divided into q substates: $\mathrm{IE}(k)$, $1 \leqslant k \leqslant q$. Here, "$k$" can be viewed as the number of informative data in the channel. The state of a module can be defined directly from the states of input and output channels: A module is in the stall state (SS) if any input channel is in the ED state or any output channel is in the SR state, otherwise, it is in the normal state (NS). Note that a module is in the SS state if and only if this module will stall in the next timestamp.

The initial state of output channels of relay stations is ED, and all other channels are in the IE(1) state initially.

The state-transition graph for communication channels is shown in Fig. 6. The state-transition rules are as follows.

1) The next state of a channel, whose current state is $\mathrm{IE}(k)$, will still be $\mathrm{IE}(k)$ if and only if both the source and sink modules are in the NS state or both of them are in the SS state. If only the source module is in the SS state, the next state will be $\mathrm{IE}(k-1)$ or ED depending on whether $k > 1$. If only the sink module is in the SS state, the next state will be $\mathrm{IE}(k+1)$ or SR based on whether $k < q$.
2) The next state of a channel, whose current state is ED, will be ED if and only if the source module is in the SS state. Otherwise, it will change to the IE(1) state.
3) The next state of a channel, whose current state is SR, will be SR if and only if the sink module is in the SS state. Otherwise, it will change to the $\mathrm{IE}(q)$ state.

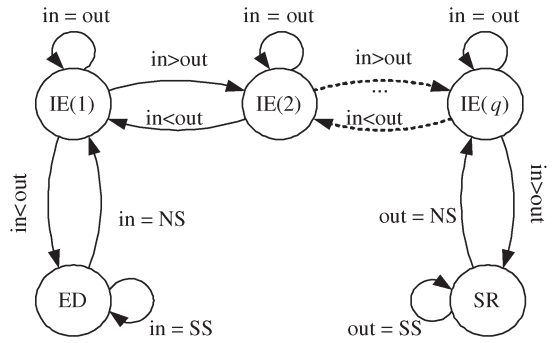

Fig. 6. State graph of a communication channel with queue size of q. "in" and "out" refer to the state of the source and sink modules, respectively. For convenience, we assume that "SS $<$ NS," therefore, in $<$ out means that the source module is in the SS state while the sink module is in the NS state. When a channel is in the ED state, the sink module is in the SS state, therefore, only the source module state should be considered. Similarly, only the sink-module state is considered for a channel in the SR state.

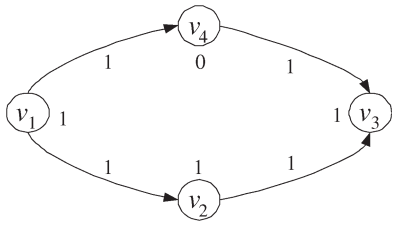

Fig. 7. BP-LIS graph for the LIS shown in Fig. 2, assuming it is a BP-LIS whose channels all have minimum queues.

A LIS that follows the above behavior can handle backpressure easily. We refer to such an LIS "BP-LIS."

We define BP-LIS graphs to capture the structure of BP-LISs as follows:

Definition 3.1: A BP-LIS graph $G_{\mathrm{B}} = (V, E, q, \alpha)$ is a weighted directed connected graph, where V is the set of all modules including circuit blocks and relay stations, $(v_i, v_j) \in E$ refers to the communication channel from module v_i to module v_j. The edge-weight function $q : E \to \mathbb{N}_0$ defines the queue size of each communication channel (v_i, v_j). The vertex-weight function $\alpha : V \to \{0, 1\}$ distinguishes relay stations and circuit blocks; $\alpha(v_i)$ is 0 if and only if the module corresponding to v_i is a relay station.

Example 3.1: Fig. 7 shows the BP-LIS graph for the LIS shown in Fig. 2 when the LIS is implemented as a BP-LIS whose channels all have minimum queues.

IV. Modeling of LISs With State Traces

In this section, we introduce "state trace" to model the behaviors and analyze the performance of general LISs (G-LISs). Here, a G-LIS is an LIS in which a module stalls in the next timestamp if one of its input channels is empty. Note that BP-LISs are a subset of G-LISs. (In fact, for LISs in [13] and [14], a B-module also stalls in the next timestamp when any of its input ports has no informative data available either from the shell queue or directly from the source B-module of that input port. Such a situation is equivalent to an empty input channel from the viewpoint of this paper. Therefore, LISs in [13] and [14] are also subsets of G-LISs. Hence, state traces can also be applied to systems in [13] and [14].)

State traces have some similarities with the "progressive traces" defined in [14]. Both of them use a sequence of numbers or symbols to capture system behaviors. However, progressive traces are defined for communication channels, whereas state traces are for modules. Progressive traces include special symbol "τ" to denote noninformative events, but all entries in state traces are integers. As we shall see later, channel state and informative data accumulated in a channel can be computed immediately from state traces. Most important, through state traces, the behaviors of LISs can be modeled formally with max-plus algebra.

In the remainder of this section, we first introduce state traces to model G-LISs and define important properties such as throughput, stability, and cyclicity, based on state traces. The performance of G-LISs is then analyzed and a throughput upper bound is proved.

A. Modeling of General-LISs

For convenience, we use the following G-LIS graphs.

Definition 4.1: A G-LIS graph $G_{\mathrm{L}} = (V, E, \alpha)$ is a directed connected graph with weighted vertices. The vertex-weight function $\alpha : V \to \{0, 1\}$ distinguishes relay stations and circuit blocks; $\alpha(v_i)$ is 0 means that the module corresponding to v_i is a relay station.

Definition 4.2: Let v_i be a module of a G-LIS $G_{\mathrm{L}}(V, E, \alpha)$, the state trace $x_i = (x_i(1), x_i(2), x_i(3), \ldots)$ of module v_i is an infinite sequence of nonnegative integers such that, for any timestamp $t \in \mathbb{N}_0$

$$x_i(t) \stackrel{\mathrm{def}}{=} \begin{cases} \alpha(v_i), & \text{if } t = 1 \\ x_i(t-1), & \text{if } t > 1 \text{ and module } v_i \text{ stalls at timestamp } t \\ x_i(t-1) + 1, & \text{otherwise.} \end{cases} \quad (4)$$

In words, $x_i(t)$ corresponds to the number of informative data generated by module v_i till timestamp t. In the first timestamp, every circuit block produces its first informative data. Therefore, $x_i(1)$ is 1 for any circuit block v_i. Relay stations are different; they can only stall before they receive informative data from input channels. Therefore, $x_i(1)$ is equal to 0 for any relay station v_i. After the first timestamp, $x_i(t)$ increases by 1 from $x_i(t-1)$ as long as module x_i does not stall at timestamp t.

With the state traces of all modules, a behavior of a G-LIS is defined as follows.

Definition 4.3: Given a G-LIS $G_{\mathrm{L}}(V, E, \alpha)$, a behavior $X \in \mathbb{R}^{|V| \times \infty}$ is a $|V| \times \infty$ matrix

$$X \stackrel{\mathrm{def}}{=} (\, x_1 \quad x_2 \quad x_3 \quad \cdots \quad x_{|V|} \,)^{\mathrm{T}} \quad (5)$$

where x_i is the state trace for module v_i. The system state at timestamp t is a vector

$$X(t) \stackrel{\mathrm{def}}{=} (\, x_1(t) \quad x_2(t) \quad x_3(t) \quad \cdots \quad x_{|V|}(t) \,)^{\mathrm{T}}. \quad (6)$$

TABLE I
SYSTEM STATUS IN THE FIRST FIVE TIMESTAMPS FOR THE BP-LIS IN FIG. 7

	$x_i(t)$				Channel State			
t	1	2	3	4	a	b	c	d
1	1	1	1	0	IE	IE	IE	ED
2	2	2	1	1	IE	SR	IE	IE
3	3	2	2	2	SR	IE	IE	IE
4	3	3	3	3	IE	IE	ED	IE
5	4	4	4	3	IE	IE	IE	ED

The system behavior can also be written as

$$X = (\, X(1) \quad X(2) \quad X(3) \quad \cdots).$$

Note that, given a G-LIS G_{L}, there may be many different behaviors because both system implementations (for example, without or with back-pressure) and external influences may lead to different behaviors. Here, two behaviors X and X' are called to be different if the two matrices X and X' are not the same.

Example 4.1: Table I shows the system states in the first five timestamps for the BP-LIS shown in Fig. 7 (and Fig. 2). Note that we use a BP-LIS example here because BP-LISs are a subset of G-LISs. In timestamp 1, all circuit blocks produce their first informative data, whereas the relay station stalls. The input channel d of block 3 is in state ED because its source module is a relay station. Consequently, block 3 stalls in timestamp 2 and we have $x_3(2) = x_3(1) = 1$. Therefore, the first informative data generated by block 2 in channel b is not processed, and it is stored in the queue of channel b. As a result, channel b becomes full (i.e., in state SR) in the timestamp 2. Therefore, block 2 stalls in timestamp 3 and we have $x_2(3) = x_2(2) = 2$. Table I shows the states of all modules and channels in the first five timestamps. Therefore, the system behavior X is

$$X = \begin{pmatrix} 1 & 2 & 3 & 3 & 4 & \dots \\ 1 & 2 & 2 & 3 & 4 & \dots \\ 1 & 1 & 2 & 3 & 4 & \dots \\ 0 & 1 & 2 & 3 & 3 & \dots \end{pmatrix}.$$

Note that here, we assume that there is no external influence in the first five timestamps. Consider a different situation: If there is a primary input to module 1, and this primary input is not able to supply informative data for the module at timestamp 1, module 1 stalls at timestamp 2 ($x_1(2) = x_1(1)$). The system behavior is changed to X'

$$X' = \begin{pmatrix} 1 & 1 & 2 & 3 & 4 & \dots \\ 1 & 2 & 2 & 3 & 3 & \dots \\ 1 & 1 & 2 & 2 & 3 & \dots \\ 0 & 1 & 1 & 2 & 3 & \dots \end{pmatrix}.$$

Therefore, under different external influences, a BP-LIS can have many different behaviors.

A circuit block produces its second informative data only after it receives and consumes the first informative data in each input channel. In general, a circuit block produces its $(m+1)$th informative data to its output channels in the next timestamp after it receives mth informative data from each of its input channels. A relay station produces its mth informative data after it consumes the mth informative data from its input channel. Therefore, a module, say v_j, produces its mth informative data after it consumes $(m - \alpha(v_j))$th informative data from each of its input channels. Hence, for a channel (v_i, v_j) in G-LIS $G_{\mathrm{L}}(V, E, \alpha)$, we have

$$x_i(t) \geq x_j(t) - \alpha(v_j). \tag{7}$$

Definition 4.4: Given a G-LIS $G_{\mathrm{L}}(V, E, \alpha)$, the set of informative data accumulated in a channel (v_i, v_j) at timestamp t is

$$f_{i,j}(t) \stackrel{\mathrm{def}}{=} \begin{cases} \emptyset, & \text{if } x_i(t) = x_j(t) - \alpha(v_j) \\ \{x_i(t), x_i(t)-1, \dots, \\ \quad x_j(t) - \alpha(v_j) + 1\}, & \text{otherwise.} \end{cases} \tag{8}$$

Therefore, the number of informative data in channel (v_i, v_j) at timestamp t is

$$|f_{i,j}(t)| = x_i(t) - x_j(t) + \alpha(v_j). \tag{9}$$

The throughput of a module can be computed when its state trace is given.

Definition 4.5: Given a state trace x_i for a module v_i in a G-LIS, the throughput $\vartheta(x_i)$ of module v_i is

$$\vartheta(x_i) \stackrel{\mathrm{def}}{=} \lim_{t \to \infty} \frac{x_i(t)}{t}. \tag{10}$$

The throughput, in fact, is the average number of informative data generated for each timestamp by the corresponding module. In general, it is possible that such a limit for the throughput may not be well defined for an arbitrary state trace. For an LIS without back-pressure and external influence, the results in [14] show that the above limit, or a well-defined throughput, exists. As we shall see later, state traces of a BP-LIS without external influences also have a well-defined limit. However, one can always specify for a primary input of a BP-LIS a state trace that is without a well-defined throughput (see Example 4.2). Such an input would render the throughput of the BP-LIS ill defined.

Example 4.2: This is an example state trace without well-defined throughput

$$(\underbrace{1,}_{A}\ \underbrace{2,3,}_{B}\underbrace{3,3,3,3,3,3,}_{A}\underbrace{4,5,6,7,\dots,20,21,}_{B}\dots).$$

The state trace has two types of interleaving segments: A and B. In an A-type segment, $x_i(t) = x_i(t-1)$, and in a B-type segment, $x_i(t)$ is always equal to $x_i(t-1)+1$. The length of each segment is equal to two times the total length of all segments before it. Consider the last item of any A-type segment other than the first one

$$\frac{x_i(t)}{t} = \frac{x_i(3n)}{3n} = \frac{x_i(n)}{3n} \leqslant \frac{1}{3}.$$

And for the last item of any B-type segment

$$\frac{x_i(t)}{t} = \frac{x_i(3n)}{3n} = \frac{x_i(n)+2n}{3n} \geqslant \frac{2}{3}.$$

Therefore, this state trace does not have a well-defined throughput.

In practice, however, most systems display good average characteristics when given sufficient observation time. Therefore, we assume that the limit for throughput exists in the remainder of this paper.

Definition 4.6: Given a system behavior X for a G-LIS $G_{\mathrm{L}}(V, E, \alpha)$, the system throughput $\vartheta(X)$ is defined as

$$\vartheta(X) = \min_{v_i \in V} \vartheta(x_i).$$

In practice, a system is useful only if all modules have the same throughput. Consider a behavior X where not all state traces $x_i \in X$ have the same throughput, we can always find a channel (v_i, v_j) such that $\vartheta(x_i) \neq \vartheta(x_j)$. Then

$$\lim_{t\to\infty} |f_{i,j}(t)| = \lim_{t\to\infty} \left(\left(\frac{x_i(t)}{t} - \frac{x_j(t)}{t} \right) \times t + \alpha(v_j) \right) = \infty$$

which means that an infinite number of informative data will be accumulated in channel (v_i, v_j). Clearly, no queue in practice can provide infinite capacity; hence, it is impossible for an LIS to avoid the loss of informative data and wrong behavior. An LIS without back-pressure cannot automatically guarantee the same throughput of all modules, therefore, such LISs may require "equalization," as in [14], in order to behave in a stable fashion.

Definition 4.7: A behavior X of a G-LIS $G_{\mathrm{L}}(V, E, \alpha)$ is said to be cyclic if there exist positive integers T, M, and δ such that

$$\begin{aligned} &\forall t \geq M, \\ &X(t+T) = \delta \otimes X(t) \\ &\qquad = (\, x_1(t)+\delta \quad x_2(t)+\delta \quad \cdots \quad x_n(t)+\delta \,)^{\mathrm{T}}. \end{aligned}$$

The minimum T is called the period, and X is said to be T cyclic.

Corollary 4.1: For a T cyclic behavior X, we always have that

$$\vartheta(X) = \vartheta(x_i) = \frac{\delta}{T}, \qquad \forall x_i \in X.$$

Proof: The throughput of any state trace x_i can be computed as

$$\begin{aligned} \vartheta(x_i) &= \lim_{t\to\infty} \frac{x_i(t)}{t} = \lim_{n\to\infty} \frac{x_i(n\times T+m)}{n\times T+m} \\ &= \lim_{n\to\infty} \frac{n\times\delta + x_i(m)}{n\times T+m} = \frac{\delta}{T}. \end{aligned}$$

Therefore, for any cyclic behavior, all modules have the same throughput, which is equal to the system throughput δ/T. ■

B. Performance of General LISs

We now study the performance of G-LISs. Consider the relationship between system states of two consecutive timestamps: t and $t+1$. By the definition of state traces in (4), we have

$$x_i(t+1) \leqslant x_i(t) + 1, \qquad \forall v_i \in V. \tag{11}$$

Consider a module v_i with (v_s, v_i) being its input channel. Channel (v_s, v_i) forces module i to stall at timestamp $t+1$ when it is empty at timestamp t:

$$x_i(t+1) = x_i(t), \qquad \text{if } |f_{s,i}| = x_s(t) - x_i(t) + \alpha(v_i) = 0.$$

This can be converted to

$$x_i(t+1) = x_s(t) + \alpha(v_i). \tag{12}$$

In the case when channel (v_s, v_i) is not empty

$$\begin{aligned} |f_{s,i}| = x_s(t) - x_i(t) + \alpha(v_i) \geq 1 \\ \text{or } x_i(t)+1 \leqslant x_s(t) + \alpha(v_i). \end{aligned}$$

When combined with (11)

$$x_i(t+1) \leqslant x_s(t) + \alpha(v_i). \tag{13}$$

Based on (12) and (13), we have

$$x_i(t+1) \leqslant x_s(t) + \alpha(v_i) \tag{14}$$

regardless of the state of channel (v_s, v_i).

Definition 4.8: Given a G-LIS $G_{\mathrm{L}}(V, E, \alpha)$, the system matrix $A_{\mathrm{L}} \in \mathbb{R}_{\infty}^{|V|\times|V|}$ is a $|V| \times |V|$ matrix, where each entry is determined by the following equation

$$(A_{\mathrm{L}})_{i,j} \stackrel{\mathrm{def}}{=} \begin{cases} \alpha(v_i), & \text{if } \exists (v_j, v_i) \in E \\ 1, & \text{if } i = j \\ \infty, & \text{otherwise.} \end{cases}$$

The system matrix can be represented by its precedence graph $\mathcal{G}(A_{\mathrm{L}})$ (see Section II), in which every edge corresponds to a noninfinity entry of the matrix. The noninfinity entries in the system matrix correspond to either "channel edges" or "self-edges" in the precedence graph.

Example 4.3: Consider the LIS shown in Fig. 2, the system matrix is

$$\begin{pmatrix} 1 & \infty & \infty & \infty \\ 1 & 1 & \infty & \infty \\ \infty & 1 & 1 & 1 \\ 0 & \infty & \infty & 1 \end{pmatrix}.$$

The precedence graph is shown in Fig. 8. Channel edge (v_1, v_4) with weight of 0 corresponds to $(A_{\mathrm{L}})_{4,1}$; self-edge (v_3, v_3) is from $(A_{\mathrm{L}})_{3,3}$.

Theorem 4.2: Given a G-LIS $G_{\mathrm{L}}(V, E, \alpha)$, its system matrix A_{L}, and its behavior X, for all timestamp $t \geq 1$

$$X(t+1) \leqslant A_{\mathrm{L}} \otimes X(t).$$

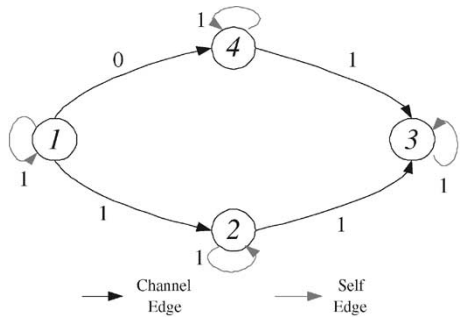

Fig. 8. Example precedence graph for the LIS in Fig. 2.

Proof: When expressed in traditional algebra, it becomes

$$x_i(t+1) \leqslant \min_{1\leqslant j\leqslant |V|}\Big((A_{\mathrm{L}})_{i,j}+x_j(t)\Big), \qquad \forall i\in\{1,2,\ldots,|V|\}.$$

This essentially summarizes (14) and (11). ■

Now, we are ready to analyze the maximum throughput that can be achieved by LISs.

Corollary 4.3: Given a G-LIS $G_{\mathrm{L}}(V,E,\alpha)$ and its system matrix A_{L}, for any behavior X of the LIS, its throughput $\vartheta(X)$ is at most $\lambda^*(\mathcal{G}(A_{\mathrm{L}}))$, the minimum cycle mean of the precedence graph $\mathcal{G}(A_{\mathrm{L}})$.

Proof: Consider the precedence graph $\mathcal{G}(A_{\mathrm{L}})$ and its critical cycle $C=\{i_1,i_2,\ldots,i_n,i_1\}$. Based on Theorem 4.2

$$\begin{aligned} x_{i_2}(t+1) &\leqslant x_{i_1}(t)+(A_{\mathrm{L}})_{i_2,i_1}\\ x_{i_3}(t+2) &\leqslant x_{i_2}(t+1)+(A_{\mathrm{L}})_{i_2,i_3}\\ &\cdots\\ x_{i_1}(t+n) &\leqslant x_{i_n}(t+n-1)+(A_{\mathrm{L}})_{i_n,i_1}. \end{aligned}$$

Adding the preceding inequalities

$$x_{i_1}(t+n)\leqslant x_{i_1}(t)+w(C).$$

Then

$$x_{i_1}(t)\leqslant \left\lfloor \frac{t}{|C|}\right\rfloor \times w(C)+r(t)$$

where

$$r(t)=x_{i_1}\left(t-\left\lfloor\frac{t}{|C|}\right\rfloor\times|C|\right)<|C|.$$

Therefore, the system throughput is given by

$$\begin{aligned}\vartheta(X)\leqslant\vartheta(x_{i_1})&=\lim_{t\to\infty}\frac{x_{i_1}(t)}{t}\leqslant\lim_{t\to\infty}\frac{\left\lfloor\frac{t}{|C|}\right\rfloor\times w(C)+r(t)}{\left\lfloor\frac{t}{|C|}\right\rfloor\times|C|+r(t)}\\ &=\frac{w(C)}{|C|}=\lambda^*(\mathcal{G}(A_{\mathrm{L}})).\end{aligned}$$

■

There are two kinds of cycles in the precedence graph corresponding to the two types of edges in the precedence graph.

1) A system cycle is a cycle with only channel edges.
2) A self-loop is a cycle with only one self-edge.

The mean weight of a self-loop is always 1. For a system cycle $C=\{i_1,i_2,\ldots,i_n,i_1\}$, $w(C)=\sum_{k=1}^{n}\alpha(v_{i_k})$ is the number of circuit blocks in the cycle, and $|C|$ is the total number of modules including both circuit blocks and relay stations.

V. Performance Analysis of BP-LISs

In this section, we analyze the performance of BP-LISs. As a subset of G-LISs, BP-LISs follow all properties of G-LISs shown in Section IV. BP-LISs have additional properties: For a BP-LIS $G_{\mathrm{B}}(V,E,q,\alpha)$, the queue of any channel (v_i,v_j) has a specific queue capacity $q(v_i,v_j)$, and when a channel (v_i,v_j) is full at timestamp t, the source module v_i stalls at timestamp $t+1$.

A channel (v_i,v_j) is full (or in state SR) when the number of informative data in the channel is $q(v_i,v_j)+1$, and the source module v_i will stall to avoid further increase of informative data when the channel (v_i,v_j) is full. Therefore, the number of informative data in the channel (v_i,v_j) can never be larger than $q(v_i,v_j)+1$

$$|f_{i,j}(t)|=x_i(t)-x_j(t)+\alpha(v_j)\leqslant q(v_i,v_j)+1. \tag{15}$$

Theorem 5.1: All modules in a BP-LIS $G_{\mathrm{B}}(V,E,q,\alpha)$ always have the same throughput.

Proof: Consider any behavior X of a BP-LIS $G_{\mathrm{B}}(V,E,q,\alpha)$. From (7) and (15), we have for any channel $(v_p,v_q)\in E$

$$-\alpha(v_q)+x_q(t)\leqslant x_p(t)\leqslant x_q(t)+q(v_p,v_q)+1-\alpha(v_q).$$

Therefore

$$\begin{aligned}\vartheta(x_q)&=\lim_{t\to\infty}\frac{x_q(t)-\alpha(v_q)}{t}\leqslant\vartheta(v_p)\\ &\leqslant\lim_{t\to\infty}\frac{x_q(t)+q(v_p,v_q)+1-\alpha(v_q)}{t}=\vartheta(x_q)\end{aligned}$$

or simply, $\vartheta(x_p)=\vartheta(x_q)$. As BP-LIS G_{B} is connected, the throughputs of all modules are equal. ■

Theorem 5.1 shows that BP-LISs, unlike LISs without backpressure in [14], do not require "equalization" to make the throughput of all modules equal.

A. System Matrices of BP-LISs

Again, we consider the relationship between system states of two consecutive timestamps to study the performance of BP-LISs. Consider an output channel (v_i,v_d) of module v_i. When it is full at timestamp t

$$\begin{aligned}x_i(t+1)=x_i(t)&=|f_{i,d}(t)|+x_d(t)-\alpha(v_d)\\ &=x_d(t)+q(v_i,v_d)+1-\alpha(v_d).\end{aligned}\tag{16}$$

When channel (v_i, v_d) is not full

$$|f_{i,d}(t)| = x_i(t) - x_d(t) + \alpha(v_d) \leqslant q(v_i, v_d).$$

Then

$$x_i(t+1) \leqslant x_i(t) + 1 \leqslant x_d(t) + q(v_i, v_d) + 1 - \alpha(v_d). \tag{17}$$

Combining (16) and (17), we have

$$x_i(t+1) \leqslant x_d(t) + q(v_i, v_d) + 1 - \alpha(v_d) \tag{18}$$

regardless of the channel state.

Definition 5.1: Given a BP-LIS $G(V, E, q, \alpha)$, the system matrix $A_{\mathrm{B}} \in \mathbb{R}_{\infty}^{|V|\times|V|}$ is a $|V| \times |V|$ matrix, where each entry is determined as follows:

$$(A_{\mathrm{B}})_{i,j} \stackrel{\mathrm{def}}{=} \begin{cases} \alpha(v_i), & \text{if } (v_j, v_i) \in E \\ q(v_i, v_j) + 1 - \alpha(v_j), & \text{if } (v_i, v_j) \in E \\ & \text{and } (v_j, v_i) \notin E \\ 1, & \text{if } i = j \\ \infty, & \text{otherwise.} \end{cases}$$

Similar to the system matrix A_{L} for G-LISs, the system matrix A_{B} of a BP-LIS regulates the relationship between $X(t)$ and $X(t+1)$. Generally, for a channel (v_i, v_j), there are two related entries in the system matrix, corresponding to (14) and (18). There is one exception when there are two channels (v_i, v_j) and (v_j, v_i) with opposite directions between modules v_i and v_j. In such a case, $(A_{\mathrm{B}})_{i,j}$ and $(A_{\mathrm{B}})_{j,i}$ is equal to $\alpha(v_i)$ and $\alpha(v_j)$, respectively. Consider $(A_{\mathrm{B}})_{j,i}$, which constrains on $x_j(t+1) - x_i(t)$. From (14), and considering channel (v_i, v_j), we have

$$x_j(t+1) \leqslant x_i(t) + \alpha(v_j). \tag{19}$$

And from (18), and considering channel (v_j, v_i), we have

$$x_j(t+1) \leqslant x_i(t) + q(v_j, v_i) + 1 - \alpha(v_i). \tag{20}$$

But (20) is always implied by (19) because

$$\alpha(v_j) \leqslant 1 \leqslant q(v_j, v_i) + 1 - \alpha(v_i).$$

Therefore, we can set $(A_{\mathrm{B}})_{j,i}$ to $\alpha(v_j)$ and ignore (20).

Like the system matrix A_{L} of a G-LIS, the system matrix A_{B} for a BP-LIS can also be represented by its precedence graph $\mathcal{G}(A_{\mathrm{B}})$, in which each edge corresponds to a noninfinity entry of the matrix. The precedence graph $\mathcal{G}(A_{\mathrm{B}})$ for a BP-LIS has one additional kind of edges compared with the precedence graph $\mathcal{G}(A_{\mathrm{L}})$ for a G-LIS:

Definition 5.2: Given a BP-LIS $G_{\mathrm{B}}(V, E, \alpha, q)$, edge (i, j) in the precedence graph $\mathcal{G}(A_{\mathrm{B}})$ is a "mirror edge" if there exists channel $(v_j, v_i) \in E$ and $(A_{\mathrm{B}})_{j,i}$ is equal to $q(v_j, v_i) + 1 - \alpha(v_i)$.

It can be viewed that the precedence graph $\mathcal{G}(A_{\mathrm{B}})$ of a BP-LIS is obtained from the precedence graph $\mathcal{G}(A_{\mathrm{L}})$ of the corresponding G-LIS by adding a mirror edge for each channel edge.

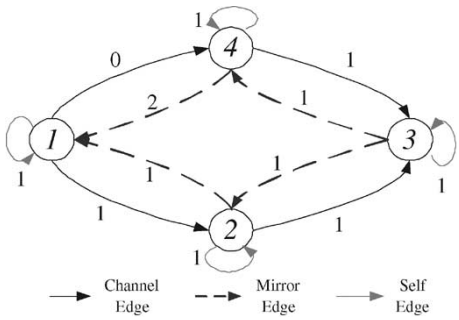

Fig. 9. Example precedence graph for the BP-LIS shown in Fig. 7. Compared with Fig. 8, for the corresponding G-LIS, this graph has one additional mirror edge with opposite direction for each channel edge.

Example 5.1: Consider a BP-LIS shown in Fig. 7; each channel of the BP-LIS has a minimum queue, the system matrix is

$$\begin{pmatrix} 1 & 1 & \infty & 2 \\ 1 & 1 & 1 & \infty \\ \infty & 1 & 1 & 1 \\ 0 & \infty & 1 & 1 \end{pmatrix}.$$

The precedence graph is shown in Fig. 9.

Theorem 5.2: Given a BP-LIS $G_{\mathrm{B}}(V, E, q, \alpha)$ and its system matrix A_{B}, for any behavior X

$$X(t+1) \leqslant A_{\mathrm{B}} \otimes X(t).$$

Proof: This is basically a summary of (11), (14), and (18). ■

B. Intrinsic Performance of BP-LISs

We now study the behaviors of BP-LISs without external influence. In other words, all system primary inputs can always supply the required data in time and all primary output would not generate SRs. We refer to this system behavior, which is completely determined by its structure, as the "intrinsic behavior," and denote the behavior by $\hat{X}$. The corresponding throughput $\vartheta(\hat{X})$ is called the "intrinsic throughput."

Theorem 5.3: Let A_{B} be the system matrix of a BP-LIS $G_{\mathrm{B}}(V, E, q, \alpha)$, the intrinsic behavior $\hat{X}$ satisfies

$$\hat{X}(t+1) = A_{\mathrm{B}} \otimes \hat{X}(t), \qquad \forall t \in \mathbb{N}_0.$$

Proof: As there are no external influences for the intrinsic behavior, a module stalls at timestamp $t+1$ if and only if there exists inside the system an empty input channel $(|f_{j,i}(t)| = 0)$ or a full output channel $(|f_{i,k}(t)| = q(v_i, v_k) + 1)$ at timestamp t. Equivalently, for the intrinsic behavior $\hat{X}$ of a BP-LIS $G_{\mathrm{B}}(V, E, q, \alpha)$, we have (21), seen at the bottom of the next page.

Therefore, when the BP-LIS graph $G_{\mathrm{B}}(V, E, q, \alpha)$ of a system is given, its intrinsic behavior is completely determined based on (21).

We now prove Theorem 5.3 by mathematical induction.

Base Case: Consider $t = 1$, or prove that $\hat{X}(2) = A_{\mathrm{B}} \otimes \hat{X}(1)$.

For any channel (v_p, v_q)

$$\begin{aligned}|f_{p,q}(1)| &= \hat{x}_p(1) - \hat{x}_q(1) + \alpha(v_q) \\ &= \alpha(v_p) - \alpha(v_q) + \alpha(v_q) = \alpha(v_p).\end{aligned}$$

Therefore, the channel cannot be full in the first timestamp because

$$|f_{p,q}(1)| = \alpha(v_p) < 2 \leqslant q(v_p, v_q) + 1.$$

Based on (21), we have

$$\forall v_i \in V, \hat{x}_i(2) = \begin{cases} \hat{x}_i(1), & \text{if } \exists (v_j, v_i) \in E, \text{ s.t. } \alpha(v_j) = 0 \\ \hat{x}_i(1) + 1, & \text{otherwise.} \end{cases} \tag{22}$$

This means that a module stalls at the second timestamp if and only if at least one of its input channels has a relay station as its source module.

Now consider $(A_{\mathrm{B}} \otimes \hat{X}(1))$, based on the definition of system matrix A_{B}

$$\begin{aligned}\left(A_{\mathrm{B}} \otimes \hat{X}(1)\right)_i &= \bigoplus_{j=1}^{|V|} (A_{\mathrm{B}})_{i,j} \otimes \hat{x}_j(1) \\ &= \left(\bigoplus_{(v_k, v_i) \in E} (A_{\mathrm{B}})_{i,k} \otimes \hat{x}_k(1) \right) \\ &\quad \oplus \left(\bigoplus_{(v_i, v_m) \in E} (A_{\mathrm{B}})_{i,m} \otimes \hat{x}_m(1) \right) \\ &\quad \oplus (1 \otimes \hat{x}_i(1)).\end{aligned} \tag{23}$$

Clearly, the three parts of the right side correspond to the three kinds of noninfinity values in the system matrix A_{B} (see Definition 5.1).

For each item in the first part

$$(A_{\mathrm{B}})_{i,k} \otimes \hat{x}_k(1) = \alpha(i) \otimes \alpha(k) = \hat{x}_i(1) \otimes \alpha(k).$$

For each item of second part

$$\begin{aligned}(A_{\mathrm{B}})_{i,m} \otimes \hat{x}_m(1) &= (q(v_i, v_m) + 1 - \alpha(v_m)) \otimes \alpha(v_m) \\ &= q(v_i, v_m) + 1 \geqslant 2 \geqslant 1 \otimes \hat{x}_i(1).\end{aligned}$$

The rightmost side of this inequality is exactly the third part in (23). Therefore, the second part can be ignored. Hence

$$\begin{aligned}\left(A_{\mathrm{B}} \otimes \hat{X}(1)\right)_i &= \left(\bigoplus_{(v_k, v_i) \in E} \hat{x}_i(1) \otimes \alpha(k) \right) \oplus (\hat{x}_i(1) \otimes 1) \\ &= \hat{x}_i(1) \otimes \left(\left(\bigoplus_{(v_k, v_i) \in E} \alpha(k) \right) \oplus 1 \right).\end{aligned}$$

The right-hand side of the preceding equation is exactly the expression in max-plus algebra for the right-hand side of (22).

Induction Step: Assume that $\hat{X}(t) = A_{\mathrm{B}} \otimes \hat{X}(t-1)$, prove that $\hat{X}(t+1) = A_{\mathrm{B}} \otimes \hat{X}(t)$.

Based on the definition of state traces

$$\hat{X}(t-1) \leqslant \hat{X}(t) \leqslant 1 \otimes \hat{X}(t-1).$$

Hence

$$\begin{aligned}\hat{X}(t) &= A_{\mathrm{B}} \otimes \hat{X}(t-1) \leqslant A_{\mathrm{B}} \otimes \hat{X}(t) \leqslant A_{\mathrm{B}} \otimes \left(1 \otimes \hat{X}(t-1)\right) \\ &= 1 \otimes A_{\mathrm{B}} \otimes \hat{X}(t-1) = 1 \otimes \hat{X}(t).\end{aligned}$$

Equivalently

$$\hat{x}_i(t) \leqslant \left(A_{\mathrm{B}} \otimes \hat{X}(t)\right)_i \leqslant \hat{x}_i(t) + 1, \qquad \forall v_i \in V.$$

Therefore, $(A_{\mathrm{B}} \otimes \hat{X}(t))_i$ is either $\hat{x}_i(t)$ or $\hat{x}_i(t) + 1$. We consider the following two cases.

Case 1) When $(A_{\mathrm{B}} \otimes \hat{X}(t))_i$ is equal to $\hat{x}_i(t)$, based on Theorem 5.2

$$\left(A_{\mathrm{B}} \otimes \hat{X}(t)\right)_i = \hat{x}_i(t) \leqslant \hat{x}_i(t+1) \leqslant \left(A_{\mathrm{B}} \otimes \hat{X}(t)\right)_i.$$

Then, $\hat{x}_i(t+1) = (A_{\mathrm{B}} \otimes \hat{X}(t))_i$.

Case 2) When $(A_{\mathrm{B}} \otimes \hat{X}(t))_i$ is equal to $\hat{x}_i(t) + 1$

$$(A_{\mathrm{B}})_{i,j} \otimes \hat{x}_j(t) = (A_{\mathrm{B}})_{i,j} + \hat{x}_j(t) \geqslant \hat{x}_i(t) + 1.$$

Equivalently

$$\hat{x}_i(t) - \hat{x}_j(t) \leqslant (A_{\mathrm{B}})_{i,j} - 1, \qquad \forall v_j \in V. \tag{24}$$

Consider any output channel $(v_i, v_j) \in E$ of module v_i, we have

$$\begin{aligned}|f_{i,j}(t)| &= \hat{x}_i(t) - \hat{x}_j(t) + \alpha(v_j) \leqslant (A_{\mathrm{B}})_{i,j} - 1 + \alpha(v_j) \\ &= q(v_i, v_j) < q(v_i, v_j) + 1.\end{aligned}$$

Therefore, any output channel of the module v_i is not full.

$$\forall v_i \in V, t \geqslant 1, \hat{x}_i(t+1) = \begin{cases} \hat{x}_i(t), & \text{if } \exists (v_j, v_i) \in E, \text{ s.t. } |f_{j,i}(t)| = 0, \\ & \quad \text{or } \exists (v_i, v_k) \in E, \text{ s.t. } |f_{i,k}(t)| = q(v_i, v_k) + 1 \\ \hat{x}_i(t) + 1, & \text{otherwise.} \end{cases} \tag{21}$$

Similarly, consider any input channel (v_k, v_i). From (24)

$$\hat{x}_i(t) - \hat{x}_k(t) \leqslant (A_{\mathrm{B}})_{i,k} - 1.$$

Then we have

$$|f_{k,i}(t)| = \hat{x}_k(t) - \hat{x}_i(t) + \alpha(v_i) \geqslant 1 - (A_{\mathrm{B}})_{i,k} + \alpha(v_i) = 1 > 0. \tag{25}$$

Hence, it is not empty. Based on (21)

$$\hat{x}_i(t+1) = \hat{x}_i(t) + 1 = \left(A_{\mathrm{B}} \otimes \hat{X}(t)\right)_i.$$

Combining the two cases 1) and 2), we have $\hat{x}_i(t+1) = (A_{\mathrm{B}} \otimes \hat{X}(t))_i$, $\forall i \in \{1, 2, \ldots, |V|\}$, or equivalently, $\hat{X}(t+1) = A_{\mathrm{B}} \otimes \hat{X}(t)$.

Based on the principle of mathematical induction, $\hat{X}(t+1) = A_{\mathrm{B}} \otimes \hat{X}(t)$, $\forall t \in \mathbb{N}_0$. ■

Theorem 5.3 can be used to compute the intrinsic throughput of BP-LISs.

Corollary 5.4: The intrinsic behavior $\hat{X}$ of any BP-LIS $G_{\mathrm{B}}(V, E, q, \alpha)$ is cyclic, and the throughput is $\lambda^*(\mathcal{G}(A_{\mathrm{B}}))$, where A_{B} is the system matrix, and $\lambda^*(\mathcal{G}(A_{\mathrm{B}}))$ is the minimum cycle mean of the precedence graph $\mathcal{G}(A_{\mathrm{B}})$.

Proof: Based on Theorem 5.3, we have

$$\begin{aligned}\hat{X}(t) &= A_{\mathrm{B}} \otimes \hat{X}(t-1) = A_{\mathrm{B}} \otimes A_{\mathrm{B}} \otimes \hat{X}(t-2) \\ &= \cdots = A_{\mathrm{B}}^{\otimes(t-1)} \otimes \hat{X}(1), \qquad \forall t > 1.\end{aligned}$$

Based on the definition of system matrix A_{B}, for each edge (v_i, v_j) in $G_{\mathrm{B}}(V, E, q, \alpha)$, there must be two corresponding edges (i, j) and (j, i) with opposite directions in $\mathcal{G}(A_{\mathrm{B}})$. In addition, BP-LIS graph $G_{\mathrm{B}}(V, E, q, \alpha)$ is connected. Therefore, $\mathcal{G}(A_{\mathrm{B}})$ is strongly connected.

From Theorem 2.1, $\exists M, T \in \mathbb{N}$, such that $\forall t > M$

$$\begin{aligned}\hat{X}(t+T) &= A_{\mathrm{B}}^{\otimes(t+T-1)} \hat{X}(1) = (\lambda^*)^{\otimes T} \otimes A_{\mathrm{B}}^{\otimes t-1} \otimes \hat{X}(1) \\ &= (\lambda^*)^{\otimes T} \otimes \hat{X}(t) = (T \times \lambda^*) \otimes \hat{X}(t)\end{aligned}$$

where λ^* refers to $\lambda^*(\mathcal{G}(A_{\mathrm{B}}))$. Therefore, the intrinsic behavior is cyclic. Based on Corollary 4.1, the intrinsic throughput is $\vartheta(\hat{X}) = (T \times \lambda^*)/T = \lambda^*$. ■

Example 5.2: Considered the BP-LIS in Fig. 7 and its precedence graph in Fig. 9. The minimum cycle mean $\lambda^*(\mathcal{G})$ is 3/4 and the critical cycle is $\{v_1, v_4, v_3, v_2, v_1\}$. The intrinsic behavior is

$$\hat{X} = \begin{pmatrix} 1 & 2 & 3 & 3 & 4 & 5 & \ldots \\ 1 & 2 & 2 & 3 & 4 & 5 & \ldots \\ 1 & 1 & 2 & 3 & 4 & 4 & \ldots \\ 0 & 1 & 2 & 3 & 3 & 4 & \ldots \end{pmatrix}.$$

We can see that $\hat{x}_i(t+4) = \hat{x}_i(t) + 3$, $\forall t \geqslant 1$ and $1 \leqslant i \leqslant 4$. The intrinsic behavior has a period of 4 and the throughput is 3/4, which is equal to the minimum cycle mean of the precedence graph $\mathcal{G}(A_{\mathrm{B}})$.

There are three kinds of cycles in the precedence graph corresponding to the three types of edges in the precedence graph.

1) A system cycle is a cycle with only channel edges.
2) A self-loop is a cycle with only a self-edge.
3) An extended cycle is a cycle with mirror edges.

While system cycles and self-loops are also in the precedence graph of the corresponding LIS system matrix A_{L}, extended cycles exist only in $\mathcal{G}(A_{\mathrm{B}})$. Therefore, we have

$$\lambda^*\left(\mathcal{G}(A_{\mathrm{B}})\right) \leqslant \lambda^*\left(\mathcal{G}(A_{\mathrm{L}})\right).$$

The mean weight of an extended cycle is not only determined by the distribution of relay stations in the extended cycle, but also affected by the queue sizes of the channels corresponding to mirror edges. Based on Definition 5.1, the weight of a mirror edge (v_j, v_i) is $(q(v_i, v_j) + 1 - \alpha(v_j))$. Therefore, the mean weight of an extended cycle can always be increased by increasing the sizes of channel queues corresponding to its mirror edges. Therefore, with proper queue size assignment, $\lambda^*(\mathcal{G}(A_{\mathrm{B}}))$ will not be limited by extended cycles.

Corollary 5.5: Given a G-LIS $G_{\mathrm{L}}(V, E, \alpha)$, there is always a corresponding BP-LIS $G_{\mathrm{B}}(V, E, q, \alpha)$ whose intrinsic throughput $\vartheta(\hat{X}) = \lambda^*(\mathcal{G}(A_{\mathrm{L}}))$, where A_{L} is the system matrix of LIS G_{L}.

Proof: This can be proved constructively by an algorithm that constructs such a BP-LIS. For any given BP-LIS $G_{\mathrm{B}}(V, E, q_0, \alpha)$, as long as the critical cycle in the precedence graph is an extended cycle, the mean weight of this cycle is increased to a value not less than $\lambda^*(\mathcal{G}(A_{\mathrm{L}}))$ by increasing the size of the channel queue corresponding to one mirror edge in the extended cycle. The process continues until the critical cycle in the BP-LIS graph becomes a system cycle or a self-loop. The intrinsic throughput of the final BP-LIS becomes equal to $\lambda^*(\mathcal{G}(A_{\mathrm{L}}))$. ■

This shows the efficiency of the proposed BP-LISs: Their intrinsic throughput can always reach the highest throughput achievable by latency-insensitive design with proper queue sizing.

Example 5.3: Consider the precedence graph in Fig. 9. The critical cycle $\{v_1, v_4, v_3, v_2, v_1\}$ is an extended cycle because (v_2, v_1) and (v_3, v_2) are mirror edges. There are no system cycles in the precedence graph. Therefore, the maximum throughput is 1. This throughput can be realized by increasing the queue size of channel (v_1, v_2) or (v_2, v_3) to 2. If queue size of (v_2, v_3) is increased to 2, the intrinsic behavior becomes

$$\hat{X} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & \ldots \\ 1 & 2 & 3 & 4 & 5 & 6 & \ldots \\ 1 & 1 & 2 & 3 & 4 & 5 & \ldots \\ 0 & 1 & 2 & 3 & 4 & 5 & \ldots \end{pmatrix}$$

whose intrinsic throughput is 1.

Example 5.4: Fig. 10 shows a more complex BP-LIS example, which corresponds to the structure of a Motion Pictures Expert Group (MPEG)-2 video decoder from [14]. All channels have minimum queues initially. Let A_{B} be the system matrix of the BP-LIS, and A_{L} be the system matrix for corresponding

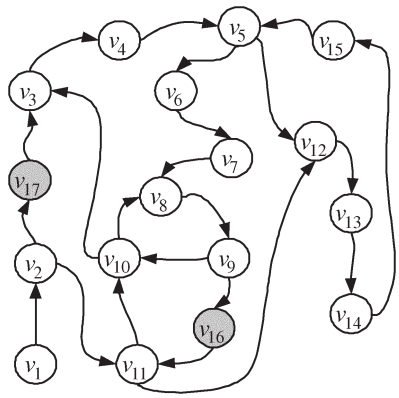

Fig. 10. BP-LIS graph of an MPEG-2 video encoder, whose structure is from [14]. All edge weights are 1, which means all channels have minimum queue. All vertices other than v_{16} and v_{17} have weight 1, which means v_{16} and v_{17} are relay stations.

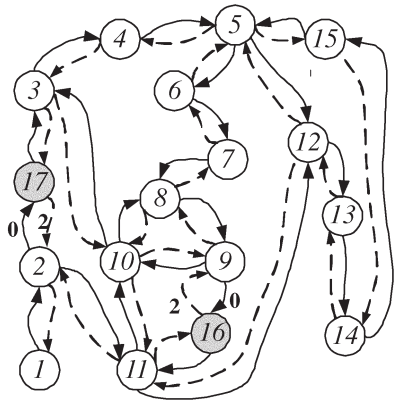

Fig. 11. Precedence graph $\mathcal{G}(A_{\mathrm{B}})$ for the BP-LIS in Fig. 10. All self-edges are omitted. Edge weight is 1 if it is not noted in the graph. The precedence graph $\mathcal{G}(A_{\mathrm{L}})$ for the corresponding G-LIS can be acquired by removing all mirror edges in this graph.

G-LIS. There are three cycles in $\mathcal{G}(A_{\mathrm{L}})$, which include at least one vertex corresponding to a relay station

$$C_1 = (8, 9, 16, 11, 10, 8), \lambda(C_1) = \frac{4}{5}$$

$$C_2 = (8, 9, 16, 11, 10, 3, 4, 5, 6, 7, 8), \lambda(C_2) = \frac{9}{10}$$

$$C_3 = (8, 9, 16, 11, 12, 13, 14, 15, 5, 6, 7, 8), \lambda(C_3) = \frac{10}{11}.$$

All these are channel cycles and can be mapped to cycles in the G-LIS or BP-LIS graph. The critical cycle is C_1, and the minimum cycle mean of $\mathcal{G}(A_{\mathrm{L}})$ is 4/5. Fig. 11 shows the precedence graph $\mathcal{G}(A_{\mathrm{B}})$ for the BP-LIS shown in Fig. 10. Because $\mathcal{G}(A_{\mathrm{L}})$ is a subgraph of $\mathcal{G}(A_{\mathrm{B}})$, all cycles in $\mathcal{G}(A_{\mathrm{L}})$ also exist in $\mathcal{G}(A_{\mathrm{B}})$. However, $\mathcal{G}(A_{\mathrm{B}})$ has additional extended cycles. The extended cycles with cycle mean less than $\lambda^*(\mathcal{G}(A_{\mathrm{L}}))$ are

$$C_1' = (2, 17, 3, 10, 9, 16, 11, 2), \lambda(C_1') = \frac{5}{7}$$

$$C_2' = (9, 16, 11, 10, 9), \lambda(C_2') = \frac{3}{4}$$

$$C_3' = (2, 17, 3, 10, 8, 9, 16, 11, 2), \lambda(C_3') = \frac{6}{8}.$$

Therefore, the intrinsic throughput is $\lambda^*(\mathcal{G}(A_{\mathrm{B}})) = 5/7$, which is about 10.7% smaller than $\lambda^*(\mathcal{G}(A_{\mathrm{L}}))$. Now, consider the three extended cycles one by one. For C_1', (3,10), (10,9), and (11,2) are mirror edges corresponding to channels (v_{10}, v_3), (v_9, v_{10}), and (v_2, v_{11}), respectively. Increasing the queue size of any of the three channels to 2 leads to $\lambda(C_1')$ equal to 6/7, which is consequently larger than $\lambda^*(\mathcal{G}(A_{\mathrm{L}}))$. Suppose we increase the queue size of channel (v_{10}, v_3) to 2. Because (3, 10) is also a mirror edge in extended cycle C_3', the cycle mean of C_3' is also increased to 7/8, which is larger than $\lambda^*(\mathcal{G}(A_{\mathrm{L}}))$. The intrinsic throughput becomes 3/4, which is limited by C_2'. Similarly, suppose that we increase the queue size of (v_9, v_{10}). $\lambda(C_2')$ becomes 1. Now, channel cycle C_1 becomes the critical cycle; no further queue sizing can improve the performance. The intrinsic throughput is increased from 5/7 to 4/5 at the cost of two additional buffers in channel queues.

An alternative queue-sizing solution can achieve the same intrinsic throughput. If we increase the queue size of channel (v_2, v_{11}) and (v_9, v_{10}) to 2, the intrinsic throughput of the BP-LIS also becomes 4/5.

Now let us consider the equalization of the corresponding LIS without back-pressure in [14]. The LIS graph includes three strongly connected components

$$S_1 = \{v_1\}$$

$$S_2 = \{v_2\}$$

$$S_3 = \{v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}\}.$$

Note that B-relay stations do not appear directly in these components because only B-circuit blocks have corresponding vertices in the "LIS graph," as defined in [14]. The throughput of the three components are 1, 1, and 4/5, respectively. Because the three components connects in series, it is not necessary to really equalize their throughput. Instead, the effect of equalization can be realized as long as the first component S_1 has the smallest throughput. To slow down a component like S_1, which has only one vertex and has no cycle, it is proposed in [14] that a self-loop with extra B-relay stations is added to the only vertex. For S_1, adding one B-relay station on the self-loop will reduce the throughput to 1/2. Hence, the equalization results in a final throughput of 1/2. This throughput is less than that of the corresponding BP-LIS either before or after queue sizing.

To achieve the same throughput, there often exist many queue-sizing solutions. The flexibilities may help to reduce the total area cost of channel queues. Most important, it is possible to achieve high performance improvement even when there are stringent area constraints. This paper focuses on the performance analysis of LISs. For channel-queue-sizing algorithms, the reader may refer to [36].

BP-LISs always have correct system behaviors regardless of the primary input/output activities. However, the primary inputs or outputs do affect the system behavior and performance. It is

easy to see that the intrinsic behavior $\hat{X}$ defines an upper bound of any behavior X of a BP-LIS

$$X(t) \leqslant \hat{X}(t) \qquad \forall t \in \mathbb{N}_0.$$

Obviously, the intrinsic throughput defines the maximum performance that a BP-LIS can achieve under any input/output combinations.

VI. Conclusion and Future Works

The behaviors of LISs are formally modeled and studied based on max-plus algebra. We proposed BP-LIS, an implementation of the latency-insensitive protocol that can provide robust communication because of back-pressure. It has also been shown that the intrinsic throughput of BP-LISs with proper queue sizing can always reach the best performance achievable by general latency-insensitive design.

In the future, we will study behaviors of BP-LISs under external influences. Mixed-timing interfaces for LISs have been proposed in [37] and extends latency-insensitive design into multiclock domains. Modeling and performance analysis of LISs in multiclock domains will also be considered.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions that helped to improve this manuscript.

References

[1] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integr. VLSI J.*, vol. 21, no. 1–2, pp. 1–94, Nov. 1996.

[2] D. Matzke, "Will physical scalability sabotage performance gains?" *IEEE Computer*, vol. 30, no. 9, pp. 37–39, Sep. 1997.

[3] R. Lu, G. Zhong, C.-K. Koh, and K.-Y. Chao, "Flip-flop and repeater insertion for early interconnect planning," in *Design, Automation Test Europe Conf.*, Paris, France, 2002, pp. 690–695.

[4] P. Cocchini, "Concurrent flip-flop and repeater insertion for high performance integrated circuits," in *Proc. Int. Conf. Computer Aided Design*, San Jose, CA, 2002, pp. 268–273.

[5] S. Hassoun, C. J. Alpert, and M. Thiagarajan, "Optimal buffered routing path constructions for single and multiple clock domain systems," in *Proc. Int. Conf. Computer Aided Design*, San Jose, CA, 2002, pp. 247–253.

[6] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, no. 1, pp. 87–116, 1991.

[7] C. Lin and H. Zhou, "Retiming for wire pipelining in system-on-chip," in *Proc. Int. Conf. Computer Aided Design*, San Jose, CA, 2003, pp. 215–220.

[8] C. Chu, F. Y. Young, K. Y. Tong, and S. Dechu, "Retiming with interconnect and gate delay," in *Proc. Int. Conf. Computer Aided Design*, San Jose, CA, 2003, pp. 221–226.

[9] R. Lu and C.-K. Koh, "Interconnect planning with local area constrained retiming," in *Design, Automation Test Europe Conf.*, Munich, Germany, 2003, pp. 442–447.

[10] S. Hassoun and C. Ebeling, "Architectural retiming: Pipelining latency-constrained circuits," in *Proc. Design Automation Conf.*, Las Vegas, NV, 1996, pp. 708–713.

[11] ——, "Using precomputation in architecture and logic resynthesis," in *Proc. Int. Conf. Computer Aided Design*, San Jose, CA, 1998, pp. 316–323.

[12] L. P. Carloni, K. L. McMillan, and A. L. Sangiovanni-Vincentelli, "Theory of latency-insensitive design," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 9, pp. 1059–1076, Sep. 2001.

[13] L. P. Carloni, K. L. McMillan, A. Saldanha, and A. L. Sangiovanni-Vincentelli, "A methodology for correct-by-construction latency insensitive design," in *Proc. Int. Conf. Computer Aided Design*, San Jose, CA, 1999, pp. 309–315.

[14] L. P. Carloni and A. L. Sangiovanni-Vincentelli, "Performance analysis and optimization of latency insensitive systems," in *Proc. Design Automation Conf.*, Los Angeles, CA, 2000, pp. 361–367.

[15] ——, "Coping with latency in SOC design," *IEEE Micro, Special Issue Systems on Chip*, vol. 22, no. 5, pp. 24–35, 2002.

[16] F. L. Baccelli, G. Cohen, and G. J. Olsder, *Synchronization and Linearity: An Algebra for Discrete Event Systems*. New York: Wiley, 1992.

[17] S. Hauck, "Asynchronous design methodologies: An overview," *Proc. IEEE*, vol. 83, no. 1, pp. 69–93, Jan. 1995.

[18] A. Mathur, A. Dasdan, and R. K. Gupta, "Rate analysis for embedded systems," *ACM Transact. Des. Automat. Electron. Syst.*, vol. 3, no. 3, pp. 408–436, 1998.

[19] S. Malik, M. Martonosi, and Y.-T. S. Li, "Static timing analysis of embedded software," in *Proc. Design Automation Conf.*, Anaheim, CA, 1997, pp. 147–152.

[20] T.-Y. Yen and W. Wolf, "Performance estimation for real-time distributed embedded systems," in *Proc. IEEE Int. Conf. Computer Design*, Austin, TX, 1995, pp. 64–69.

[21] T. Zhou, X. Hu, and E. H.-M. Sha, "A probabilistic performance metric for real-time system design," in *Proc. Int. Conf. Hardware/Software Codesign*, Rome, Italy, 1999, pp. 90–94.

[22] A. Kalavade and P. Moghe, "A tool for performance estimation of networked embedded end-systems," in *Proc. Design Automation Conf.*, San Francisco, CA, 1998, pp. 257–262.

[23] R. Marculescu, A. Nandi, L. Lavagno, and A. L. Sangiovanni-Vincentelli, "System-level power/performance analysis of portable multimedia systems communicating over wireless channels," in *Proc. Int. Conf. Computer Aided Design*, San Jose, CA, 2001, pp. 207–214.

[24] C. V. Ramamoorthy and G. S. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets," *IEEE Trans. Softw. Eng.*, vol. 6, no. 5, pp. 440–449, Sep. 1980.

[25] S. M. Burns, "Performance analysis and optimization of asynchronous circuits," Ph.D. thesis, Comput. Sci. Dept., Calif. Inst. Technol., Pasadena, 1991.

[26] T. Lee, "A general approach to performance analysis and optimization of asynchronous circuits," Ph.D. dissertation, Comput. Sci. Dept., Calif. Inst. Technol., Pasadena, 1995.

[27] P. Kudva, G. Gopalakrishnan, E. Brunvand, and V. Akella, "Performance analysis and optimization of asynchronous circuits," in *Proc. IEEE Int. Conf. Computer Design*, Cambridge, MA, 1994, pp. 221–224.

[28] A. Xie and P. A. Beerel, "Symbolic techniques for performance analysis of timed systems based on average time separation of events," in *Proc. Int. Symp. Advanced Research Asynchronous Circuits Systems*, Eindhoven, The Netherlands, 1997, pp. 64–75.

[29] ——, "Performance analysis of asynchronous circuits and systems using stochastic timed Petri nets," in *Proc. 2nd Workshop Hardware Design Petri Nets*, Williamsburg, VA, 1999, pp. 35–62.

[30] R. M. Karp, "A characterization of the minimum cycle mean in a digraph," *Discrete Math.*, vol. 23, no. 3, pp. 309–311, 1978.

[31] A. Dasdan and R. Gupta, "Faster maximum and minimum mean cycle algorithms for system performance analysis," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 17, no. 10, pp. 889–899, Oct. 1998.

[32] G. Cohen, D. Dubois, J.-P. Quadrat, and M. Viot, "A linear system-theoretic view of discrete event processes and its use for performance evaluation in manufacturing," *IEEE Trans. Autom. Control*, vol. 30, no. 3, pp. 210–220, Mar. 1985.

[33] J. P. Quadrat, M. Akian, G. Cohen, S. Gaubert, and M. Viot, "Max-plus algebra and applications to system theory and optimal control," in *Proc. Int. Congress Mathematicians*, Zurich, Switzerland, 1994, pp. 1502–1511.

[34] S. Gaubert, "Rational series over dioids and discrete event systems," in *Proc. 11th Conf. Analysis Optimization Systems: Discrete Event Systems*, Sophia Antipolis, France, 1994, pp. 247–256.

[35] G. Cohen, S. Gaubert, and J.-P. Quadrat, "Max-plus algebra and system theory: Where we are and where to go now," *Annu. Rev. Control*, vol. 23, no. 1, pp. 207–219, 1999.

[36] R. Lu and C.-K. Koh, "Performance optimization of latency insensitive systems through buffer queue sizing of communication channels," in *Proc. Int. Conf. Computer Aided Design*, San Jose, CA, 2003, pp. 227–231.

[37] T. Chelcea and S. M. Nowick, "Robust interfaces for mixed-timing systems with application to latency-insensitive protocols," in *Proc. Design Automation Conf.*, Las Vegas, NV, 2001, pp. 21–26.

Ruibing Lu received the B.E. and M.E. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1997 and 2000, respectively. He received the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, in 2004.

Currently, he is a Senior R&D Engineer at Synopsys, Inc., Mountain View, CA. His research interests include multivoltage and interconnect-centric design optimization, on-chip communication analysis, design and optimization.

Cheng-Kok Koh (S'92–M'98) received the B.S. degree with first class honors and the M.S. degree, both in computer science, from the National University of Singapore in 1992 and 1996, respectively. He received the Ph.D. degree in computer science from University of California, Los Angeles in 1998.

Currently, he is an Associate Professor of Electrical and Computer Engineering at Purdue University, West Lafayette, IN. His research interests include physical design of high-performance low-power very large scale integration (VLSI) circuits, with an emphasis on VLSI interconnect layout optimization.

Dr. Koh received the Lim Soo Peng Book Prize for Best Computer Science Student from the National University of Singapore in 1990, and the Tan Kah Kee Foundation Postgraduate Scholarship in 1993 and 1994. He received the GTE Fellowship and the Chorafas Foundation Prize from the University of California at Los Angeles in 1995 and 1996, respectively. He received the ACM Special Interest Group on Design Automation (SIGDA) Meritorious Service Award and Distinguished Service Award in 1998, the Chicago Alumni Award from Purdue University in 1999, the National Science Foundation CAREER Award in 2000, and the ACM/SIGDA Distinguished Service Award in 2002.