Matrix Approach to Deadlock-Free Dispatching in Multi-Class Finite Buffer Flowlines

A Gürel[1], S. Bogdan[2] F. L. Lewis[3]

[1] Department of Electrical and Electronic Engineering, The Eastern Mediterranean University, Famagusta, via Mersin 10 TURKEY. E-mail: gurel@eenet.ee.emu.edu.tr
[2]Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, HR-10000 Zagreb, CROATIA. E-mail: stjepan.bogdan@fer.hr
[3]Automation and Robotics Research Institute, The University of Texas at Arlington, 7300 Jack Newell Blvd. S., Ft Worth, TX 76118-7115, USA. E-mail: flewis@controls.uta.edu

Abstract. For finite-buffer manufacturing systems, the major stability issue is 'deadlock', rather than 'bounded-buffer-length stability.' The paper introduces the concept of 'system deadlock,' defined rigorously in Petri net terms, and system operation with uninterrupted part-flow is characterised in terms of the absence of this condition. For a large class of finite-buffer multi-class re-entrant flowline systems an analysis of 'circular waits' yields necessary and sufficient conditions for the occurrence of 'system deadlock'. This allows the formulation of a maximally permissive one-step-look-ahead deadlock-avoidance control policy for dispatching jobs, while maximizing the percent utilisation of resources. The result is a generalized kanban dispatching strategy, which is more general than the standard multi-class last buffer first serve (LBFS) dispatching strategies for finite buffer flowlines that typically under-utilize the resources. The problem of computational complexity associated with Petri net applications is overcome by using certain sub-matrices of the PN incidence matrix. Computationally efficient matrix techniques are given for implementing the deadlock-free dispatching policy.

1. Introduction

In *flexible manufacturing systems* (FMS) [2] resource sharing is ubiquitous: a given resource may be common to the production processes of several part-types (*parallel sharing*), and/or may be used multiple times during the production process of a given part-type (*sequential sharing* or *reentrance*). A key role in job routing/dispatching is played by the FMS *controller*, which allocates resources to perform jobs for customers or on parts. Failure by the controller to assign resources suitably during job dispatching can lead to serious performance problems. There are numerous formal job-dispatching rules, such as first-in-first-out (FIFO), first-buffer-first-serve (FBFS), last-buffer-first-serve (LBFS), earliest due date (EDD), least slack (LS), and so on [11], [13].

One fundamental question that needs to be addressed in connection with any FMS dispatching policy is whether it is *stable*. Studies of stability for FMS often focus on stability in the sense of *bounded buffer lengths* [9], [11]. However, in practice the buffer lengths are *finite*, and such stability results are inapplicable, since it is not obvious how to keep the buffer lengths below some *fixed finite value*. For finite-buffer multi-class reentrant flowline (MRF) systems [9], which constitute a large class of FMSs, the issue is stability, not in the sense of bounded buffer lengths, but in the sense of absence of *deadlock*. A flowline for a given part-class is said to be deadlocked if it holds a part that cannot complete its processing sequence. Many popular dispatching rules can result in deadlock if care is not taken (for instance, see [14]). In [11] the FBFS and LBFS policies have been shown to be stable for single-part flowlines with no buffer limits; however, both policies can cause deadlock when the buffer lengths are finite. In a finite-buffer system, any dispatching policy for uninterrupted part flow essentially has to take into account the s*tructure* of the interaction between jobs and resources. Several results based on such a structural approach may be discovered in [1], [4], [7], [8], [10], and [15]. In all of these but [7] *Petri net* (PN) formalism is used for system modeling.

This paper focuses on MRF systems without *assembly*. Based on a PN model, flowline deadlock is characterised in terms of *part-path deadlock*, and using this notion the more general condition of *system deadlock* is defined. By exploiting the structure of the job/resource interactions, it is shown that for a large class of systems, called the *regular* MRF systems, the two types of deadlock are equivalent to a condition called *circular blocking* (CB) [10]. The analysis not only generalizes the results of [3] and [15] to a much larger class of systems, but also offers a rigorous structural framework in which the problem of 'impending part flow deadlocks' mentioned in these references can be handled.

The paper also provides dynamic job-dispatching rules driven by a one-step-look-ahead deadlock-avoidance control policy. A generalized multi-part LBFS job dispatching policy is defined and shown to be deadlock-free. The proposed control scheme is maximally permissive [5] and can be viewed as a sort of generalized kanban job-dispatching strategy, including the special LBFS policies of [11]. Matrix techniques, based on certain sub-matrices of the PN incidence matrix, which come directly from industrial engineering methods, are used to obtain analytic computational formulae for deadlock analysis and avoidance. This deals with the well-known problem of complexity arising from PN applications.

2. MULTIPLE REENTRANT FLOWLINE SYSTEM MODEL

Let Π be the set of distinct types of parts produced (or customers served) by an MRF system, where each part type $k \in \Pi$ is characterized by a predetermined sequence of jobs $\mathrm{J}^k = \{\mathrm{J}_1^k, \mathrm{J}_2^k, \ldots, \mathrm{J}_{\mathrm{L}_k}^k\}$, with at least one resource used for each job. Let R denote the set of system resources, with each $\mathrm{r} \in \mathrm{R}$ a pool of multiple copies of a given resource. Since there is no assembly involved, one can uniquely associate with each job sequence J^k the operations of raw part-in, $\mathrm{J}^k_{\mathrm{in}}$, and finished product-out, $\mathrm{J}^k_{\mathrm{out}}$.

The MRF system activity is described by a PN model, called the *flowline system PN* (FPN), defined as follows. Let $N=(\mathrm{P},\mathrm{T},\mathrm{I},\mathrm{O})$ denote the FPN, where I represents the input arcs to T from P and O represents the output arcs from T to P. Given any node $\mathrm{v}\in\mathrm{P}\cup\mathrm{T}$, let ${}^\bullet\mathrm{v}$ and $\mathrm{v}^\bullet$ respectively denote the pre-set and post-set of v in the usual way, i.e., the set of nodes that have arcs to and from v, respectively. For a set of nodes $\mathrm{V}=\{\mathrm{v}_i\}$, define ${}^\bullet\mathrm{V}=\{{}^\bullet\mathrm{v}_i\}$ and $\mathrm{V}^\bullet = \{\mathrm{v}_i^\bullet\}$. Define $\mathrm{P}=\mathrm{R}\cup\mathrm{J}\cup\mathrm{J}_{\mathrm{in}}\cup\mathrm{J}_{\mathrm{out}}$, with R, J_{in}, and $\mathrm{J}_{\mathrm{out}}$ as the sets of places respectively representing the availability of resources, part arrivals and finished products, and J as the set of places representing the ongoing jobs. The set of transitions T can be partitioned as $\mathrm{T}=\cup_{k\in\Pi}\mathrm{T}^k$, where $\mathrm{T}^k=\{\mathrm{t}^k_1,\mathrm{t}^k_2,\ldots\mathrm{t}^k_{\mathrm{L}_k+1}\}$, with $\mathrm{t}^k_i = {}^\bullet\mathrm{J}^k_i = \mathrm{J}^k_{i\text{-}1}{}^\bullet$, for $i\notin\{1,\mathrm{L}_k\}$; while $\mathrm{t}^k_1={}^\bullet\mathrm{J}^k_1=\mathrm{J}^k_{\mathrm{in}}{}^\bullet$ and $\mathrm{t}^k_{\mathrm{L}_k+1}=\mathrm{J}^k_{\mathrm{L}_k}{}^\bullet={}^\bullet\mathrm{J}^k_{\mathrm{out}}$. In other words, each transition corresponds to a *decision* or *rule* for the starting and/or completion of a job, which also involves the allocation and/or release of at least one resource. Thus $\mathrm{J}^k\cup\mathrm{T}^k$ defines a distinct *part-path*, with initial place $\mathrm{J}^k_{\mathrm{in}}={}^\bullet\mathrm{t}^k_1$, with ${}^\bullet\mathrm{J}^k_{\mathrm{in}}=\varnothing$, and terminal place $\mathrm{J}^k_{\mathrm{out}}=\mathrm{t}^k_{\mathrm{L}_k+1}{}^\bullet$, with $\mathrm{J}^k_{\mathrm{out}}{}^\bullet=\varnothing$. Resource places always occur off part-paths. For any $\mathrm{r}\in\mathrm{R}$, define the *job set* $\mathrm{J(r)}$ as the set of jobs using r, and *resource loop* $\mathrm{L(r)}=\mathrm{r}\cup\mathrm{J(r)}$. Given a set of resources $\mathrm{Q}\subset\mathrm{R}$, define the job set of Q as $\mathrm{J(Q)}=\cup_{\mathrm{r}\in\mathrm{Q}}\mathrm{J(r)}$. Denote by $\mathrm{R}(\mathrm{J}^k_i)$ the resources used by job J^k_i.

An available resource or an ongoing job is indicated by *tokens* in the respective places. It is assumed that places in J_{in} are always marked (i.e., there is always a part ready to enter) and those in $\mathrm{J}_{\mathrm{out}}$ are always empty (i.e., finished product is pulled out immediately). The *marking* $m:\ \mathrm{P}\rightarrow\mathrm{Z}$, with Z as the set of nonnegative integers, gives the distribution of tokens. $\{N,m\}$ denotes the *marked* FPN. The *initial marking* m_0 represents the *idle state* (i.e., no parts in process), and $\Re(m_0)$ shows the set of all markings reachable from m_0. For a set of places S, the set marking is defined as $m(\mathrm{S})=\sum_{\mathrm{p}\in\mathrm{S}}m(\mathrm{p})$. A transition

$\mathrm{t} \in \mathrm{T}$ is said to be *dead* at m if there exists no $m' \in \Re(m)$ that enables it, with $\Re(m)$ as the set of markings reachable from m. t is said to be *job(resource)-enabled* if $m({}^\bullet\mathrm{t} \cap \mathrm{J}) > 0$ ($m({}^\bullet\mathrm{t} \cap \mathrm{R}) > 0$). A marking m is said to be dead if no $\mathrm{t} \in \mathrm{T}$ is enabled at m. A place $\mathrm{p} \in \mathrm{P}$ is said to be dead at m if $m(\mathrm{p}) = 0 = m'(\mathrm{p})$ for all $m' \in \Re(m)$.

In a PN, a siphon (trap) is a set of places $\mathrm{S} \subset \mathrm{P}$ ($\mathrm{Q} \subset \mathrm{P}$) such that ${}^\bullet\mathrm{S} \subset \mathrm{S}^\bullet$ ($\mathrm{Q}^\bullet \subset {}^\bullet\mathrm{Q}$). A key feature of a siphon (trap) is that, once its marking becomes zero (nonzero) it will never again become nonzero (zero). Thus if a siphon contains a marked trap, it will never become empty. A siphon (trap) is said to be *minimal* if it does not contain any other siphons (traps). It is easy to see that every resource loop $\mathrm{L(r)}$ is both a minimal siphon and a minimal trap and its total token load is conserved at all $m' \in \Re(m_0)$.

Subsequent analysis in this paper deals with the class of MRF systems specified below. This class is more general than the ones in [1], [3], [5] and [15]: it allows resource pools as well as generalized buffering strategies including one buffer per machine, shared buffers or machines without buffers.

Definition 1: *[Class of MRF$_1$ systems]* Define MRF$_1$ as the class of MRF systems with the FPN description N satisfying the following: (i) $\forall \mathrm{p} \in \mathrm{P}$, ${}^\bullet\mathrm{p} \cap \mathrm{p}^\bullet \neq \varnothing$; (ii) $\forall k \in \Pi$, $\mathrm{t}_1^{k\bullet} \cap \mathrm{P} \setminus \mathrm{J} = \varnothing$ and ${}^\bullet\mathrm{t}_{\mathrm{L}_k+1}^{k} \cap \mathrm{P} \setminus \mathrm{J} = \varnothing$; (iii) $\forall \mathrm{J}_i^k \in \mathrm{J}$, $|\mathrm{R}(\mathrm{J}_i^k)| = 1$ and $\mathrm{R}(\mathrm{J}_i^k) \neq \mathrm{R}(\mathrm{J}_{i+1}^k)$; (iv) $\forall \mathrm{J}_i^k \in \mathrm{J}$, $|\mathrm{J}_i^{k\bullet}| = 1$; (v) $\forall \mathrm{t}_i^k \in \mathrm{J}$, $|{}^\bullet\mathrm{t}_i^k \cap \mathrm{J}| \leq 1$; (vi) $\forall \mathrm{r} \in \mathrm{R}$, $|\mathrm{J(r)}| \geq 1$.

This means that there are no self loops, each part-path has a well-defined beginning and an end, every job requires one and only one resource with no two consequent jobs using the same resource, there are no choice jobs and no assembly jobs, and there are shared resources. Obviously in MRF$_1$ systems, for any $\mathrm{r} \in \mathrm{R}$, $\mathrm{J(r)} = \mathrm{r}^{\bullet\bullet} \cap \mathrm{J} = {}^{\bullet\bullet}\mathrm{r} \cap \mathrm{J}$ and $\mathrm{R}(\mathrm{J}_i^k) = {}^{\bullet\bullet}\mathrm{J}_i^k \cap \mathrm{R} = \mathrm{J}_i^{k\bullet\bullet} \cap \mathrm{R}$.

3. Stability For MRF$_1$ Systems — Deadlock

For finite-buffer systems, any stability analysis must address the issue of deadlock avoidance so that no part in the flowline is held up, that is every part in the flowline must come out as a finished product. The following two definitions provide a rigorous characterisation of this condition.

Definition 2: *[Part-path deadlock]* Given a system of class MRF$_1$ with $\{N, m_0\}$, part-path $k \in \Pi$, is said to be deadlocked at any $m \in \Re(m_0)$ if some job-enabled transition t_i^k is dead at m.

For any part-path $k \in \Pi$, let $\bar{\mathrm{J}}^k \subset \mathrm{J}^k$ denote a nontrivial subset of consecutive jobs. Define $M_{\bar{\mathrm{J}}^k}^m \subset \Re(m)$ to be the subset of markings reachable from m via a sequence of markings $\{m^i\}$ such that $m^i(\bar{\mathrm{J}}^k) \leq m(\bar{\mathrm{J}}^k)$, for all i (i.e., no new part goes into $\bar{\mathrm{J}}^k$). An uninterrupted flow of parts clearly means that every part in any $\bar{\mathrm{J}}^k$ can move through the rest of its job sequence without deadlocking any part-path. This requires absence of deadlock for the entire system in the sense specified below.

Definition 3: *[System deadlock]* Given a system of class MRF$_1$ with $\{N, m_0\}$, N is said to be deadlocked at any $m \in \Re(m_0)$ if either (a) some part-path is deadlocked at m, or (b) there exists some $\bar{\mathrm{J}}^k \subset \mathrm{J}^k$ with $m(\bar{\mathrm{J}}^k) > 0$, such that at every $m^i \in M_{\bar{\mathrm{J}}^k}^m$ with $m^i(\bar{\mathrm{J}}^k)=0$, some part-path is deadlocked.

Consider now a dispatching policy U for $\{N, m_0\}$, and let $M_U \subset \Re(m_0)$ denote the set of markings reachable under U. Note that U, by controlling the firing of certain transitions of N, limits the set of reachable markings to M_U. Such a policy is said to be maximally permissive if M_U is maximal with respect to certain imposed specifications [5], e.g., deadlock-freeness. A deadlock-free dispatching policy can be defined as follows.

Definition 4: *[Deadlock-free dispatching policy]* A dispatching policy U is said to be deadlock-free if N is not deadlocked at any $m \in M_U$.

It is shown in [10] that part-path deadlock is equivalent to a system condition called *circular blocking* (CB), which is a consequence of the existence of *circular wait relations*. System deadlock, on the other hand, specifies a condition in which, although there may be no CB, further dispatching of jobs will inevitably result in one (cf. 'impending part flow deadlock' in [3], and 'cyclic deadlock structure chain with key resource' in [15]). Thus, avoiding CB is necessary but generally not sufficient for a deadlock-free dispatching policy.

The notions of circular wait and circular blocking are recalled next (see [10] for further detail). For any two $\mathrm{r}_i, \mathrm{r}_j \in \mathrm{R}$, r_i is said to *wait* for r_j, denoted $\mathrm{r}_i \rightarrow \mathrm{r}_j$, if the availability of r_j is an immediate requirement for the release of r_i, i.e., if ${}^\bullet\mathrm{r}_i \cap \mathrm{r}_j{}^\bullet \neq \varnothing$. This binary relation on R is referred to as a *wait relation* and can be captured by the *wait relation graph* $\mathrm{G}_w = (\mathrm{R}, \mathrm{A})$, a digraph where R is the set of nodes, and $\mathrm{A} = \{\mathrm{a}_{ij}\}$ is the set of edges, with a_{ij} drawn if $\mathrm{r}_i \rightarrow \mathrm{r}_j$. In G_w, $\mathrm{r}_{i_1} \rightarrow \mathrm{r}_{i_2} \rightarrow \ldots \rightarrow \mathrm{r}_{i_w}$ defines an *R-path* $\mathrm{r}_{i_1} \mapsto \mathrm{r}_{i_w}$. Any set $\mathrm{C} \subset \mathrm{R}$, $|\mathrm{C}|>1$, is said to be a circular wait (CW), if for every ordered pair $\{\mathrm{r}_i, \mathrm{r}_j\} \subset \mathrm{C}$, one has $\mathrm{r}_i \mapsto \mathrm{r}_j$. Note that this is a generalization of the 'simple circuit' of [3] and [15], allowing the handling of more complex structures involving part-reentrance and resource pools. A CW C is said to be in CB if (a) $m(\mathrm{C}) = 0$; and (b) for each $\mathrm{r} \in \mathrm{C}$, $\forall\, \mathrm{p} \in \mathrm{J}(\mathrm{r})$ with $m(\mathrm{p}) \neq 0$, $\mathrm{p}^\bullet \in \mathrm{C}^\bullet$.

The following is a structural analysis of deadlock for class MRF$_1$ systems in terms CWs in CB. The proofs for the novel results are given in the appendix, and others may be discovered in [10].

Theorem 1: Given a system of class MRF$_1$ with $\{N, m_0\}$, and marking $m \in \Re(m_0)$, the following are equivalent: (i) there exists a job-enabled dead transition at m; (ii) there exists a dead resource place at m; (iii) there exists CB at m.

The theorem establishes the equivalence between part-path deadlock and CB, and hence the marking at which this occurs. This lays the ground for the characterisation of deadlock as a FPN state. A useful corollary to the theorem is as follows.

Corollary 1: Given $\{N, m_0\}$, any $m \in \Re(m_0)$ is non-dead if there is no CB at m.

Thus, if there is no CB at $m \in \Re(m_0)$, then there is at least one enabled transition. The proof of one of the main results to be stated later, namely Theorem 3, depends on this result. Now, the occurrence of CB is closely related to the special FPN structures of *critical siphon* and *critical trap*. A critical siphon (trap) is a minimal siphon (trap) that does not contain any resource loop. Note that the critical siphon defined here is similar to the siphon considered in [4], which however fails to exploit the relation between the siphons and CWs when formulating its deadlock-prevention control policies.

For constructing critical siphons and traps, consider a CW C, and transition sets, $\mathrm{T}_{\mathrm{C}}^{+} = {}^{\bullet}\mathrm{C} \setminus \mathrm{C}^{\bullet}$ and $\mathrm{T}_{\mathrm{C}}^{-} = \mathrm{C}^{\bullet} \setminus {}^{\bullet}\mathrm{C}$. Define the *siphon-job set* as $\mathrm{J}_{\mathrm{S}}(\mathrm{C}) = \mathrm{J(C)} \cap {}^{\bullet}\mathrm{T}_{\mathrm{C}}^{+}$ and the *trap-job set* as $\mathrm{J}_{\mathrm{Q}}(\mathrm{C}) = \mathrm{J(C)} \cap \mathrm{T}_{\mathrm{C}}^{-\,\bullet}$. Then the sets $\mathrm{S}_{\mathrm{C}} = \mathrm{C} \cup \mathrm{J}_{\mathrm{S}}(\mathrm{C})$ and $\mathrm{Q}_{\mathrm{C}} = \mathrm{C} \cup \mathrm{J}_{\mathrm{Q}}(\mathrm{C})$ respectively define the critical siphon and critical trap associated with C. One can now state the following for $\{N, m_0\}$ of a MRF$_1$ system [10].

Theorem 2: A CW C in N is in CB at any $m \in \Re(m_0)$ if and only if $m(\mathrm{S}_{\mathrm{C}})=0$.

This result shows that CB, and hence part-path deadlock, is equivalent to some marking (obviously not unique) that empties a specific critical siphon, namely the one associated with the CW that is in CB. Now, in keeping track of the marking of each critical siphon S_{C}, it is useful to regard each CW C as a token distribution centre, with $m(\mathrm{C})$ defined as its *kanban content*, and partition the corresponding set of token receiving places $\mathrm{J(C)}$ as follows. First, note that in general $\mathrm{J}_{\mathrm{S}}(\mathrm{C}) \cap \mathrm{J}_{\mathrm{Q}}(\mathrm{C}) = \mathrm{J}_{\mathrm{SQ}}(\mathrm{C}) \neq \varnothing$, and define $\hat{\mathrm{J}}_{\mathrm{S}}(\mathrm{C}) = \mathrm{J}_{\mathrm{S}}(\mathrm{C}) \setminus \mathrm{J}_{\mathrm{SQ}}(\mathrm{C})$ and $\hat{\mathrm{J}}_{\mathrm{Q}}(\mathrm{C}) = \mathrm{J}_{\mathrm{Q}}(\mathrm{C}) \setminus \mathrm{J}_{\mathrm{SQ}}(\mathrm{C})$ respectively as the *strictly siphon-job* and *strictly trap-job sets*, and $\mathrm{J}_{\mathrm{N}}(\mathrm{C}) = \mathrm{J(C)} \setminus (\mathrm{J}_{\mathrm{S}}(\mathrm{C}) \cup \mathrm{J}_{\mathrm{Q}}(\mathrm{C}))$ as the *neutral-job set*. The *critical subsystem* $\mathrm{J}_{\mathrm{o}}(\mathrm{C})$ associated with CW C, as defined in [10], can be then written as $\mathrm{J}_{\mathrm{o}}(\mathrm{C}) = \hat{\mathrm{J}}_{\mathrm{Q}}(\mathrm{C}) \cup \mathrm{J}_{\mathrm{N}}(\mathrm{C})$, and one has $\mathrm{J(C)} = \mathrm{J}_{\mathrm{N}}(\mathrm{C}) \cup \mathrm{J}_{\mathrm{SQ}}(\mathrm{C}) \cup \hat{\mathrm{J}}_{\mathrm{S}}(\mathrm{C}) \cup \hat{\mathrm{J}}_{\mathrm{Q}}(\mathrm{C})$, or $\mathrm{J(C)} = \mathrm{J}_{\mathrm{S}}(\mathrm{C}) \cup \mathrm{J}_{\mathrm{o}}(\mathrm{C})$, where the sets on the right hand-side in both equations are disjoint. Since $\mathrm{C} \cup \mathrm{J(C)} = \cup_{\mathrm{r} \in \mathrm{C}} \mathrm{L(r)}$, at any $m \in \Re(m_0)$ one has $m_0(\mathrm{C}) = m(\mathrm{C}) + m(\mathrm{J(C)}) = m(\mathrm{C}) + m(\mathrm{J}_{\mathrm{S}}(\mathrm{C})) + m(\mathrm{J}_{\mathrm{o}}(\mathrm{C}))$. Therefore $m(\mathrm{S}_{\mathrm{C}}) = 0$ if and only if $m(\mathrm{J}_{\mathrm{o}}(\mathrm{C})) = m_0(\mathrm{C})$; or equivalently, $m(\mathrm{S}_{\mathrm{C}}) \neq 0$ if and only if $m(\mathrm{J}_{\mathrm{o}}(\mathrm{C})) < m_0(\mathrm{C})$, i.e., the work-in-process in the critical subsystem $\mathrm{J}_{\mathrm{o}}(\mathrm{C})$ is limited above by $m_0(\mathrm{C}) - 1$. This can be done by controlling an appropriate subset of $\mathrm{T(C)} = \mathrm{C}^{\bullet} = {}^{\bullet}\mathrm{J(C)}$, i.e., the transitions that distribute the tokens in C.

Now, one can write $\mathrm{T(C)} = \mathrm{T}_{\mathrm{pre}}(\mathrm{C}) \cup \mathrm{T}_{\mathrm{pos}}(\mathrm{C}) \cup \mathrm{T}_{\mathrm{N}}(\mathrm{C})$, where the subsets are disjoint with $\mathrm{T}_{\mathrm{pre}}(\mathrm{C}) = {}^{\bullet}\mathrm{J}_{\mathrm{o}}(\mathrm{C}) \setminus \mathrm{J}_{\mathrm{o}}(\mathrm{C})^{\bullet}$, $\mathrm{T}_{\mathrm{pos}}(\mathrm{C}) = \mathrm{J}_{\mathrm{o}}(\mathrm{C})^{\bullet} \setminus {}^{\bullet}\mathrm{J}_{\mathrm{o}}(\mathrm{C})$. Note also that $\mathrm{T}_{\mathrm{pre}}(\mathrm{C}) = {}^{\bullet}\mathrm{S}_{\mathrm{C}} \setminus {}^{\bullet}\mathrm{S}_{\mathrm{C}} = {}^{\bullet}\hat{\mathrm{J}}_{\mathrm{Q}}(\mathrm{C})$, $\mathrm{T}_{\mathrm{pos}}(\mathrm{C}) = {}^{\bullet}\hat{\mathrm{J}}_{\mathrm{S}}(\mathrm{C})$, and $\mathrm{T}_{\mathrm{N}}(\mathrm{C}) = {}^{\bullet}\mathrm{J}_{\mathrm{N}}(\mathrm{C})$, so that firing any $\mathrm{t} \in \mathrm{T}_{\mathrm{pre}}(\mathrm{C})$, $\mathrm{t} \in \mathrm{T}_{\mathrm{pos}}(\mathrm{C})$ and $\mathrm{t} \in \mathrm{T}_{\mathrm{N}}(\mathrm{C})$

respectively reduces by 1, increases by 1, and leaves unaltered the token load of $\mathrm{S_C}$. This observation is significant in devising deadlock-free job-dispatching policies.

Another structure in *N*, which is relevant in deadlock analysis, is the result of a *cyclic wait relation* between two CWs, defined below.

Definition 5: *[Cyclic circular wait and key resource]* Let $\{\mathrm{C}_i, \mathrm{C}_j\}$ be such that $|\mathrm{C}_i \cap \mathrm{C}_j|=1$, and define $\mathrm{C}_i \cap \mathrm{C}_j=\{\mathrm{r_b}\}$. If $\mathrm{T_{pos}}(\mathrm{C}_i) \cap \mathrm{T_{pre}}(\mathrm{C}_j) \subset \mathrm{r_b}^\bullet$ and $\mathrm{T_{pos}}(\mathrm{C}_j) \cap \mathrm{T_{pre}}(\mathrm{C}_i) \subset \mathrm{r_b}^\bullet$, then $\{\mathrm{C}_i, \mathrm{C}_j\}$ is said to be a *cyclic CW* (CCW). If in addition $m_0(\mathrm{r_b}) = 1$, then $\mathrm{r_b}$ is called a *key resource.*

As shown below, it is expedient to separate systems with key resources from those without. So let $\{N, m_0\}$ be called *regular* if it contains no key resources, and *irregular* otherwise. Now, define M_F = $\{m \in \Re(m_0) \,|\, m(\mathrm{S_C})=0$, for some CW $\mathrm{C}\}$, i.e., the subset of reachable markings at which some CW C is in CB. Stated below is a fundamental result for regular $\{N, m_0\}$.

Theorem 3: [*Main Theorem*] If $\{N, m_0\}$ is regular, then at every $m \in \Re(m_0) \backslash M_F$, there exists at least one transition that is enabled to fire without resulting in CB.

This result says that given $\{N, m_0\}$ with no key resources, if at any marking reached there is no CB, then there always exists at least one transition that is enabled to fire without causing CB. This means that part-flow will continue provided CB can be avoided. This leads to the next fundamental result, which says that for regular systems, system deadlock is equivalent to part-path deadlock and hence to CB.

Theorem 4: [*Main Theorem*] Given $\{N, m_0\}$ regular, N has system deadlock at any $m \in \Re(m_0)$ if and only if $m \in M_F$.

Therefore for regular systems avoiding CB is a necessary and sufficient condition for avoiding system deadlock, and hence guaranteeing that the processing of every part is completed. This result generalizes a similar but much more limited one offered in [3] and [8], whereby the flowline systems under study have neither any resource pools (i.e. $m_0(\mathrm{r})=1$, all $\mathrm{r} \in \mathrm{R}$) nor any reentrance. It must be noted that when $\{N, m_0\}$ has key resources, the system may run into system deadlock several steps before any CB actually occurs. A characterisation of system deadlock for this more complex case of irregular systems can be obtained by using the structural framework that is presented in [10] and developed herein (more on this in future work).

5. Dispatching With Deadlock Avoidance

According to Theorems 2 and 4, deadlock-free dispatching for regular MRF$_1$ system requires that no critical siphon ever becomes empty. The dynamic deadlock-free dispatching strategy presented below defines a *family of generalized kanban dispatching rules*. It is a one-step-look-ahead control policy, and is maximally permissive, also maximizing the percent utilisation of resources. It is capable of dealing with the complexities arising from the 'interleaved' configuration of the MRF systems noted in [11].

Assuming that jobs are dispatched singly, the notion of *dispatching priority* is defined as follows. Given two activated (i.e., all the preconditions for their execution satisfied) jobs J_i^k and J_j^l, J_i^k is said to be dispatched with priority over J_j^l, denoted pri(J_i^k)>pri(J_j^l), if whenever both jobs are requested simultaneously J_i^k is given the preference. Given two job sets S_1 and S_2, pri(S_1)>pri(S_2) implies pri(J_i^k)>pri(J_j^l) for every $\mathrm{J}_i^k \in \mathrm{S}_1$ and $\mathrm{J}_j^l \in \mathrm{S}_2$. The foregoing analysis leads to a multi-part LBFS dispatching rule that is deadlock-free, more general than the uniform LBFS policy of [11], and is easy to implement

Theorem 5: Given $\{N, m_0\}$ regular, suppose a LBFS dispatching policy is used such that, at any m whenever a multitude of jobs $\{\mathrm{J}_i^k\}$ are activated simultaneously, they are dispatched according to the following: for every CW C such that $\{\mathrm{J}_i^k\} \cap \mathrm{J(C)} \neq \varnothing$: (i) set pri($\mathrm{J}_\mathrm{S}(\mathrm{C}) \cup \mathrm{J}_\mathrm{N}(\mathrm{C})$)>pri($\hat{\mathrm{J}}_\mathrm{Q}(\mathrm{C})$), and (ii) do not dispatch any $\mathrm{J}_i^k \in \hat{\mathrm{J}}_\mathrm{Q}(\mathrm{C})$ if $m(\hat{\mathrm{J}}_\mathrm{Q}(\mathrm{C}) + m(\mathrm{J}_\mathrm{N}(\mathrm{C})) = m_0(\mathrm{C}) - 1$. Then, deadlock will not occur.

The deadlock-free dispatching policy stated next defines the generalized kanban strategy.

Theorem 6: Given $\{N, m_0\}$ regular, any dispatching policy U is deadlock-free if and only if, for all $m \in M_U$, when assigning dispatching priorities to a multitude of simultaneously activated jobs $\{\mathrm{J}_i^k\}$, it disallows dispatching any J_i^k whenever there is some CW C such that $J_i^k \in \hat{J}_Q(C)$ and $m(\hat{J}_Q(C)+m(\hat{J}_N(C))=m_0(C)-1$.

The proofs for both of these results follow directly from Definition 4 and Theorem 4. The 'if and only if' condition of Theorem 6 shows that the control strategy is maximally permissive. Moreover, by keeping the kanban content $m(C)$ as low as possible (including zero), the WIP in the critical subsystem $J_0(C)$ is maximized, thus maximizing the percent utilisation of resources. The difference between LBFS rule of Theorem 5 and the generalized kanban strategy of Theorem 6 is that LBFS rule is applied at all times, while the kanban rule is applied for a given CW C only when $m(C)=1$. In the latter case, there exists a possibility of loading the critical subsystem (cf. FBFS rule) even until $m(C)=0$. This is not possible for the LBFS case, for which the percent resource utilization depends on the 'luck of the draw' in terms of arrival rates and actual transition firing times. Note that the deadlock-free aspect in both of the proposed policies is based only on the structural properties of the system and is therefore valid independently of the specific part arrival times and job duration/resource set-up times.

6. Matrix Computations For Dispatching Policy Implementation

Though PNs are extremely useful in modeling and analysis of manufacturing system behaviour, they do not generally lend themselves to computationally tractable algorithms. This section attempts to remedy this drawback by providing matrix techniques to determine all the aforementioned structures, which is essential for the implementation of the proposed dispatching policies. A more detailed foundation for what follows may be discovered in [10].

Given N, define the composite binary vector $p^T=[v^T \quad r^T]$, where v and r correspond to job and resource places, J and R, respectively. Then, the PN incidence matrix can be accordingly partitioned as $W=O-I=S^T-F=[S_v^T-F_v \quad S_r^T-F_r]$, where $S^T=O$ and $F=I$. The sub-matrices S_v^T, S_r^T are the output incidence matrices, and F_v, F_r are the input incidence matrices associated with the job and resource places, respectively.

To implement the policies of Theorems 5 and 6, all the CWs in N and the associated job subsets, $\mathrm{J_S(C)}$, $\mathrm{\hat{J}_Q(C)}$ and $\mathrm{J_N(C)}$ must be determined. In [10] an algorithm is given that yields the complete set $\{\mathrm{C}_i\}$ of CWs, represented by a binary matrix X, with its *i*-th column χ_i corresponding to CW C_i. The algorithm is based on the wait relation graph $\mathcal{G}_w$ and its adjacency matrix $\mathrm{G_w}$, which can be computed as $\mathrm{G_w = S_r\ F_r}$, where the matrix multiplication is in and/or algebra, and $\mathrm{S_r}$ and $\mathrm{F_r}$ are as above. For each vector χ_i, define an equal size vector χ_{is} as its projection onto the set of shared resources, i.e., χ_{is} has nonzero entries only for the *shared* resources contained in CW C_i. Then, the job subsets $\mathrm{J_S(C}_i)$ and $\mathrm{J_Q(C}_i)$ are given by the vectors $\mathrm{J_S(C}_i) = \mathrm{F_v^T[S_r^T}\chi_{is} \wedge \overline{\mathrm{F_r}\chi_i}]$, and $\mathrm{J_Q(C}_i) = \mathrm{F_v^T[F_r}\chi_{is} \wedge \overline{\mathrm{S_r^T}\chi_i}]$. One can use these to determine the other two job subsets $\mathrm{\hat{J}_Q(C}_i)$ and $\mathrm{J_N(C}_i)$ from $\mathrm{\hat{J}_Q(C}_i) = \mathrm{J_Q(C}_i) \wedge \overline{\mathrm{J_S(C}_i)}$, and $\mathrm{J_N(C}_i) = \mathrm{F_v^T S_r^T c}_i \wedge \overline{\mathrm{J_Q(C}_i) \vee \mathrm{J_S(C}_i)}$. In all of these equations the matrix operations are carried out in and/or algebra, and $\wedge$ and $\vee$ respectively denote the element-by-element matrix 'and' and 'or' operations. A proof for the result for $\mathrm{J_S(C)}$ is presented in [10]. The other three stated herein can easily be proved using a similar approach.

It is important to note that for a given system configuration this structural analysis is performed off-line and only once. Moreover, all the computations involved depend on matrix operations so that they are of polynomial complexity.

7. Conclusion

The deadlock problem in a class of finite-buffer MRF systems is discussed. The notions of part-path deadlock and system deadlock are introduced and linked to the notion of circular blocking and the related PN structures of circular waits, critical siphons and traps. For a general class of finite-buffer MRF systems without assembly, a necessary and sufficient condition for system deadlock is provided. This leads to a maximally permissive one-step-look-ahead deadlock-free dispatching control strategy. The result is a generalized kanban class of dynamic dispatching policy, which is more general than the multi-class LBFS policy of [11] and is guaranteed to not only avoid system deadlock but also provide space for maximum resource utilization. Computationally efficient matrix techniques are given for determining the structures that are required for implementing the proposed dispatching schemes. Future research will focus on applying the proposed formalism to a discrete-event simulator popular with industrial engineering applications, as well as extending the framework to deal with more complicated structures.

References

[1] Z.A. Banaszak and B.H. Krogh, "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Trans. Robotics and Automation*, vol. 6, no. 6, pp. 724-734, Dec. 1990.

[2] Buzacott and D.D. Yao, "Flexible manufacturing systems: a review of analytical models," *Management Sci*, vol. 32, no. 7, pp. 890-905, July 1986

[3] H. Cho, T.K. Kumaran, and R.A. Wysk, "Graph-theoretic deadlock detection and resolution for flexible manufacturing systems," *IEEE Trans. Robotics and Automation*, vol. 11, no. 3, pp. 413-421, 1995.

[4] J. Ezpeleta, J.M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Robotics and Automation*, vol. 11, no. 2, pp. 173-184, 1995.

[5] L. E. Holloway and B. H. Krogh, "On closed-loop liveness of discrete-event systems under maximally permissive control," *IEEE Trans. Automat. Control*, vol. 37, no. 5, pp. 692-697, 1992.

[6] F.S. Hsieh and S.C. Chang, "Dispatching-driven deadlock avoidance controller synthesis for flexible manufacturing systems," *IEEE Trans. Robotics and Automation*, vol. 10, no 2, pp. 317-327, April 1994.

[7] M.D. Jeng and F. DiCesare, "Synthesis using resource control nets for modeling shared-resource systems," *IEEE Trans. Robotics and Automation*, vol. 11, no 3, pp. 317-327, 1995.

[8] T.K. Kumaran, W. Chang, N. Cho, R.A. Wysk, "A structured approach to deadlock detection, avoidance, and resolution in flexible manufacturing systems," *Int. J. Prod. Res.*, vol. 32, no. 10, pp. 2361-2379, 1994.

[9] P.R. Kumar and S.P. Meyn, "Stability of queueing networks and scheduling policies," *IEEE Trans. Automat. Control*, vol. 40, no. 2, pp. 251-260, 1995.

[10] F.L. Lewis, A. Gürel, S. Bogdan, A. Doğanalp and O.C. Pastravanu, "Analysis of deadlock and circular waits using a matrix model for discrete event manufacturing systems," *Automatica*, 1998.

[11] S.H. Lu and P.R. Kumar, "Distributed scheduling based on due dates and buffer priorities," *IEEE Trans. Automat. Control*, vol. 36, no. 12, pp.1406-1416, Dec. 1991.

[12] T. Murata, "Petri nets: properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541-580, 1989.

[13] S.S. Panwalker and W. Iskander, "A survey of scheduling rules," *Operations Research*, vol. 26, no. 1, pp. 45-61, 1977.

[14] T.I. Seidman, "'First come first serve' is unstable!," Univ. Maryland Baltimore County, Tech. Report, 1993.

[15] K.Y. Xing, B.S. Hu and H.X Chen, "Deadlock avoidance policy for Petri-net modelling of flexible manufacturing systems with shared resources," *IEEE Trans. Automat. Control*, vol. 41, no. 2, pp. 289-295, 1991.

[16] M.C. Zhou, F. DiCesare, A.D. Desrochers, "A hybrid methodology for synthesis of Petri net models for manufacturing systems," *IEEE Trans. Robotics and Automation*, vol. 8, no. 3, pp. 350-361, Jun. 1992.