Slack Matching Quasi Delay-Insensitive Circuits

Piyush Prakash, Alain J. Martin
Department of Computer Science
California Institute of Technology
Pasadena, CA 91125 CA

Abstract— **Slack matching is an optimization that determines the amount of buffering that must be added to each channel of a slack elastic asynchronous system in order to reduce its cycle time to a specified target. We present two methods of expressing the slack matching problem as a mixed integer linear programming problem. The first method is applicable to systems composed of either full-buffers or half-buffers but not both. The second method is applicable to systems composed of any combination of full-buffers and half-buffers.**

I. Introduction

Quasi delay-insensitive(QDI) circuits are typically described as collections of independent processes that share values exclusively through message passing. Instead of a global clock, local handshakes are used for synchronization. It has been observed that the cycle time of an asynchronous system can be greater than that of the slowest module in the system running on its own. Systems are often designed with a target cycle time, τ_0. *Slack matching* is a technique to reduce a system's cycle time by inserting buffering into communication channels. Slack matching is only guaranteed to be safe on systems that are *slack elastic*, i.e., systems such that an arbitrary amount of buffering can be added to any communication channel without affecting the correctness of the system [5]. We study the problem of adding buffers to a slack elastic system to reduce the overall cycle time of the system to τ_0.

Slack matching has been compared to the *retiming* problem in synchronous design. While slack matching has been shown to be NP-complete [2], retiming for optimal throughput can be solved in polynomial time [3].

We restrict our attention to slack matching systems consisting of processes that use each communication channel exactly once per cycle. Furthermore, we assume that the communication actions are implemented using four phase handshakes.

The *static slack* of a pipeline is the maximum number of messages that can be inserted into the pipeline, with none being removed. If a pipeline of n instances of a process has static slack $\frac{n}{2}$, the process is said to be a *half-buffer*. If a pipeline of n instances of a process has static slack n, the process is said to be a *full-buffer*.

Let p be a pipeline, with $p.in$ being its input channel and $p.out$ being its output channel. Let p operate in an environment that keeps the number of messages in p constant. Lines [4] and Williams [10] have shown that the throughput(reciprocal of cycle time) of p varies with the number of messages, m, in p. Lines [4] showed that a plot of throughput of p versus m resembles either a triangle or trapezoid. Figure 1 shows an example of such a plot. The plot can be divided into three regions based upon its slope.

1) Each pipeline stage has some delay between the start of a handshake on its input and the start of the corresponding handshake on an output. If the cycle time of p, with m messages is τ, then the mean delay between two messages in p must be less than p's cycle time, τ. Increasing m reduces this mean delay, allowing p to operate at a higher throughput. This corresponds to the region where the slope is positive.
2) If a pipeline stage does not contain its static slack number of messages, it can acknowledge events on its input channels without waiting for an acknowledgment on its output channel. However, as m is increased, some pipeline stages will contain a number of messages equal to their static slack. Such stages introduce stalls in the pipeline, waiting for an acknowledgment on the output channel before producing an acknowledgment on the input channel. When this happens, increasing m increases the number of pipeline stages that stall in this manner. Each stall sequences events that could occur concurrently, were m to be smaller. Thus increasing m reduces the system's throughput. This corresponds to the region where the slope is negative.
3) Each pipeline stage must also have some internal limits to its cycle time. This corresponds to the region of the plot where the slope is zero. If 1 and 2 always constrain the p's cycle time to be above the internal cycle time of all pipeline stages in p, then the plot is a triangle. However, if 1 and 2 do not limit p's cycle time to be above the internal cycle time of all pipeline stages, then p can only operate as fast as the stage with the greatest internal cycle time and the graph is a trapezoid.

Let $d_{\min}(p.in, p.out, \tau_0)$ and $d_{\max}(p.in, p.out, \tau_0)$ be respectively the minimum and maximum number of messages that pipeline p can contain while operating with cycle time τ_0 or less. The interval $[d_{\min}(p.in, p.out, \tau_0), d_{\max}(p.in, p.out, \tau_0)]$ is the *dynamic slack* of p at cycle time τ_0.

A. Motivation

Consider the system shown in figure 2, which has reconvergent fanout. Let the system be such that when a message is inserted into pipeline A, a message must also be inserted into pipeline B. Similarly, when a message is removed from pipeline A, a message is also removed from pipeline B. Thus, there is a fixed relationship between the number of mess

Proceedings of the 12th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'06)
0-7695-2498-2/06 $20.00 © 2006 IEEE

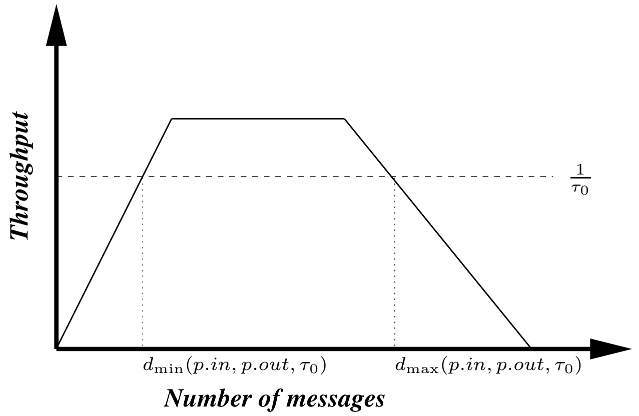

Fig. 1. Throughput of a pipeline versus number of messages

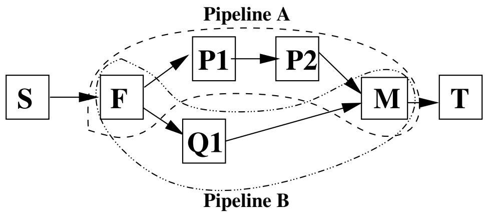

Fig. 2. System with reconvergent fanout.

the two pipelines. Let A and B both be initialized with no messages. When the system operates at cycle time τ_0, the number of messages in pipelines A and B must be within their respective dynamic slacks at τ_0. If the dynamic slacks do not intersect, then the cycle time of one of the pipelines is greater than τ_0, hence the system's cycle time must also be greater than τ_0. Adding buffers to a pipeline p increases $d_{\min}(p.in, p.out, \tau_0)$ and $d_{\max}(p.in, p.out, \tau_0)$ thus additional buffers can be added to either A or B until the dynamic slack at τ_0 of the two pipelines intersect.

Figure 3 shows how the cycle time of a ring of half-buffers, that contains one message, varies with the number of buffers on the ring. Slack matching does not change the number of messages on a ring, it can only increase the number of buffers on a ring r. Let $r.a$ be a channel of r. Increasing the number of buffers on r raises $d_{\max}(r.a, r.a, \tau_0)$ and $d_{\min}(r.a, r.a, \tau_0)$. The number of messages is dictated by the algorithm the system implements and is assumed fixed. Consider a ring, r, with m messages on the ring. If m is within r's dynamic slack at τ_0, then there is no need to slack match the ring, it already operates at the target cycle time. If m is greater than $d_{\max}(r.a, r.a, \tau_0)$, buffers can be added to r until $m \in [d_{\min}(r'.a, r'.a, \tau_0), d_{\max}(r'.a, r'.a, \tau_0)]$ where r' is a ring obtained by adding buffers to r. Thus, in this case slack matching can be used to improve the ring's cycle time. If m is less than $d_{\min}(r.a, r.a, \tau_0)$, slack matching cannot be used to improve the cycle time of the ring since adding buffers will only increase $d_{\min}(r.a, r.a, \tau_0)$ and $d_{\max}(r.a, r.a, \tau_0)$.

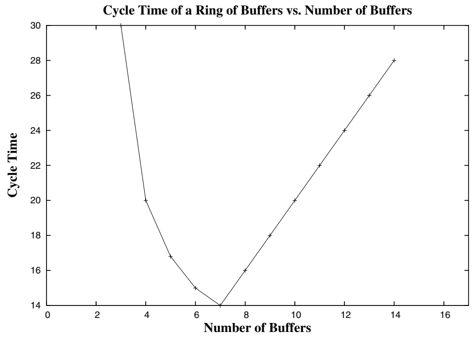

Fig. 3. Relationship between cycle time of a ring and the number of buffers on the ring.

B. Outline

This paper presents two methods of slack matching a system by solving a mixed integer linear program (MILP). The systems are described as handshaking expansion(HSE), with the HSE annotated with delays. The HSE notation used in this paper is described in the appendix. *Event-Rule* (ER) systems [1] constructed from the HSE are analyzed to determine the system's cycle time. Section II, reviews some basic results about ER systems. In section III, we state sufficient conditions under which a system composed of a specified class of half-buffers, has cycle time less than or equal to the target, τ_0. In section IV, we show how to generate a polynomial sized set of constraints that must be satisfied in order for the system to be slack matched. In section V, we present results from the application of this method to control units in two asynchronous microprocessors. In section VI, we express the problem of slack matching systems composed of full-buffers, half-buffers and slack-zero buffers as one of satisfying a set of linear equations with a subset of the variables being restricted to the integers. When the cost function is linear in the number of slack matching buffers added, mixed integer linear programming (MILP) solvers can be used to slack match a system. Section VII relates our results to prior work. We conclude in section VIII with a discussion of future work.

II. Event-Rule Systems

In the following sections, we will be analyzing the ER system that corresponds to a collection of HSE to determine a system's cycle time. In this section we review the definition of ER systems [1] and state the results about ER systems that are used in the analysis.

An ER system is a pair (E, R) where E is a set of *events* and R is a set of rules that constrain the timings of the events. Each rule of a general ER system, $r \in R$, is of the form $e \xrightarrow{\alpha} f$, where $e \in E$ and $f \in E$ are respectively the *source* and *target* of r and $\alpha \in [0, \infty)$ is the *delay* of r. The set of *sources* of an event f is $\{e | e \xrightarrow{\alpha} f \in R\}$. Similarly, the set of *targets* of an event e is $\{f | e \xrightarrow{\alpha} f \in R\}$. Whilst E and R may be infinite, the set of sources of any event e must be finite.

A *timing function* of an ER system is any function $t : E \rightarrow [0, \infty)$ such that $t(f) \geq t(e) + \alpha \quad \forall e \xrightarrow{\alpha} f \in R$. The *timing simulation*, $\hat{t}$, of an ER system is the timing function such that for any other timing function t, $\hat{t}(e) \leq t(e) \quad \forall e \in E$.

A. *Repetitive ER Systems*

Oftentimes, the behavior of a system of unbounded size can be described by a finite structure. Consider the set of events E generated from a finite set E' by $E = E' \times \mathbb{N}$. The elements of E' are referred to as *transitions*. Each event $(u, i) \in E$ is the i^{th} occurrence of transition u.

The set of rules R can be generated from a finite set of 4-tuples R' such that $r' = (u, v, \alpha, \epsilon) \in R'$ where $R' \subseteq E' \times E' \times [0, \infty) \times \mathbb{Z}$. r' is written as $(u, i - \epsilon) \xrightarrow{\alpha} (v, i)$. u and v are respectively the source and target transitions of r'. α is the delay between the $(i - \epsilon)^{th}$ occurrence of transition u and the i^{th} occurrence of transition v. ϵ is said to be the *occurrence index offset* of r'. The rule set, R, is generated by instantiating (an infinite number of times) each element of R'. In order to ensure that both the source and target of each rule is in E, it is required that $i \geq \epsilon$. The pair (E', R') is said to be a *repetitive* ER system, and (E, R) the general ER system generated from the repetitive ER system.

A linear timing function, $\bar{t}$ is a timing function such that for each event $e = (u, i)$, $\bar{t}(u, i) = x_u + i \cdot p_u$ where x_u is the *offset* of transition u and p_u its period. The *collapsed constraint graph*, G', of (E', R'), is the directed graph such that each node corresponds to a transition and each edge corresponds to an element of R'. Burns [1] shows that if two nodes, u and v, lie on the same cycle in G', then $p_u = p_v$. If the collapsed constraint graph of a repetitive ER system is strongly connected, all nodes have the same period, p.

The *minimum period linear timing function* of a repetitive ER system is the linear timing function that minimizes p. The minimum value of p can be shown to be a number arbitrarily close to the average time between successive occurrences of a transition u. This value of p is said to be the cycle time of the repetitive ER system. Burns [1] shows that the cycle time of a repetitive ER system can be determined from its collapsed constraint graph as

$$p = \max_{\text{all cycles } c} \left\{ \frac{\text{sum of delays on } c}{\text{sum of occurrence index offsets on } c} \right\}. \quad (1)$$

Burns further showed that in maximizing equation (1), only *simple* cycles(ones that do not have sub-cycles) need to be considered.

B. *Pseudo-repetitive ER Systems*

Whilst many systems can be modeled as repetitive ER systems, many systems of interest consist of a finite set of initial events followed by a set of repeated transitions. Such systems are modeled as *pseudo-repetitive* ER systems. A pseudo-repetitive ER system is a 6-tuple $(E_0, E_1, E'_0, E'_1, R_0, R'_1)$ where E_0 is a finite set of initial events, E'_0 a finite set of initial transitions, E_1 an infinite set of repeated events, E'_1 a finite set of repeated transitions, R_0 a finite set of of initial rules and R'_1 a finite set of repeated rules. Note that $E_0 \subset E'_0 \times \mathbb{N}$ and $E_1 \subseteq E'_1 \times \mathbb{N}$. The elements of R_0 take the form of rules in a general ER system, and their source must be an initial event. The elements of R'_1 are of the form of rules in a repetitive ER system. Both the source and the target of a repeated rule must be repeated transitions. The general ER system that corresponds to a pseudo-repetitive ER system is constructed by setting $E = E_0 \cup E_1$ and letting R be the union of R_0 and the rules of R'_1 instantiated in such a manner that both the target and source of the rule are in E_1.

Burns [1] shows that the cycle time of a pseudo-repetitive ER system is approximated by that of the repetitive system (E'_1, R'_1).

Example 1: Figure 4 shows the collapsed constraint graph of the repetitive part of the pseudo-repetitive ER system that describes the HSE:

$$\begin{aligned} PCHB \equiv\ & lo\uparrow;\ [ri];\ ro\downarrow;\ [\neg li];\ lo\downarrow; \\ & *[[\neg ri \wedge li];\ ro\uparrow;\ lo\uparrow;\ [ri];\ ro\downarrow;\ [\neg li];\ lo\downarrow] \end{aligned}$$

All edges have an occurrence index offset of 0 unless they are marked with a rectangular box which denotes an occurrence index offset of 1.

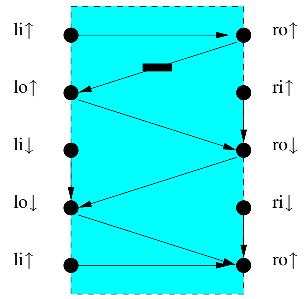

Fig. 4. Constraint graph of PCHB

III. Sufficient Conditions for Slack Matching Systems of Half-Buffers

In this section, we state sufficient conditions that guarantee that the cycle time of a system composed of half-buffers is at or below the target cycle time, τ_0. Rather than consider the collapsed constraint graph of the entire system, we represent the system as a *process graph*. First we define a process graph. Next we constrain the HSEs of the processes that comprise the system. We then label paths in the collapsed constraint graph of a process satisfying the aforementioned restrictions and state assumptions about the delays and occurrence index offsets of these paths. Given these assumptions, we state sufficient conditions on the process graph of the system to guarantee that the system's cycle time is at most τ_0. These conditions ensure that each pipeline in the system can simultaneously contain a number of messages within its dynamic slack.

A process graph $G = (V, E)$ is a directed graph where each vertex represents a process, and each edge a channel between two processes. An edge $e = (u, v)$ denotes a channel that is an output channel of process u and input channel of v. The *source* of edge $e = (u, v)$, $\text{src}(e)$, is defined to be u and the *sink*, $\text{snk}(e)$, is defined to be v. A *path* in a graph consists of a sequence of one or more distinct edges $\{e_i\}$, such that $\text{src}(e_{i+1}) = \text{snk}(e_i)$. The length of path p, denoted as $|p|$, is the number of edges that comprise the path. For any path p, the source of p, $\text{src}(p)$, is $\text{src}(e_0)$ and the sink of p, $\text{snk}(p)$, is $\text{snk}(e_{|p|-1})$. A *cycle* is a path p such that $\text{src}(p) =$ s

0-7695-2498-2/06 $20.00 © 2006 IEEE

An *undirected path* between vertices a and b is a sequence of $n \geq 1$ distinct edges, $\{e_i\}$ such that there exists a sequence of $n+1$ vertices $\{v_i\}$ with the property that for all edges e_i, either $\mathrm{src}(e_i) = v_i \wedge \mathrm{snk}(e_i) = v_{i+1}$ or $\mathrm{snk}(e_i) = v_i \wedge \mathrm{src}(e_i) = v_{i+1}$. Furthermore $v_0 = a$ and $v_n = b$. Let $\mathcal{S}(u)$ denote the set of edges e such that either $e = u$ or there exists an undirected path in G between $\mathrm{snk}(u)$ and $\mathrm{src}(e)$ that traverses neither u nor e. Let $\mathcal{K}(u)$ denote the set of edges e such that there is an undirected path in G between $\mathrm{snk}(u)$ and $\mathrm{snk}(e)$ that traverses neither u nor e.

We only consider systems composed of processes such that their repetitive portion is an implementation of one of following reshufflings, presented in Lines [4].

$PCHB \equiv *[[\neg ri \wedge li]\, ro{\uparrow};\, lo{\uparrow};\, [ri];\, ro{\downarrow};\, [\neg li];\, lo{\downarrow}]$
$WCHB \equiv *[[\neg ri \wedge li];\, ro{\uparrow};\, lo{\uparrow};\, [ri \wedge \neg li];\, ro{\downarrow};\, lo{\downarrow}]$
$B1 \equiv *[[\neg ri \wedge li];\, ro{\uparrow};\, lo{\uparrow};\, [ri \wedge \neg li];\, lo{\downarrow};\, ro{\downarrow}]$
$B4 \equiv *[[\neg ri \wedge li];\, ro{\uparrow};\, lo{\uparrow};\, [ri];\, ro{\downarrow},\, ([\neg li];\, lo{\downarrow})]$
$B5 \equiv *[[\neg ri \wedge li];\, ro{\uparrow};\, lo{\uparrow};\, [ri \wedge \neg li];\, ro{\downarrow},\, lo{\downarrow}]$

If a process has more than two channels, we assume that the channels of the process can be partitioned into two disjoint sets $\mathcal{L}$ and $\mathcal{R}$ such that the projection of the process's HSE onto the variables implementing any pair of channels $L^i \in \mathcal{L}$ and $R^j \in \mathcal{R}$ is a half buffer with one of the aforementioned reshufflings. The variables implementing L^i correspond the variables li and lo in these reshuffling, and the variables implementing R^j correspond to the variables ri and ro. The channels in $\mathcal{L}$ are said to be input channels of the process, and the channels in $\mathcal{R}$ are output channels. We adopt the convention that the four phase handshake on channel L^j is implemented by the variables li^j and lo^j, where lo^j is the variable assigned to by the process and li^j is variable that is only read by the process. Similarly, the variables ri^j and ro^j implement the communication channel R^j.

In order to determine the occurrence index offsets in the repetitive ER system, we need to specify the initial state of the system. We assume that each process, P, can be written as a straight line handshaking expansion [1] of the form $P \equiv S; *[T]$ such that the following conditions hold:

- All assignments occur at most once in S and exactly once in T.
- The projection of T onto the statements in S is S.
- If an assignment, a, appears in S, then S also contains all assignments a' such that $a; a'$ appears in the projection of T onto the set $\{a, a'\}$.
- If a wait on a variable w, $[f(w)]$ appears in S, then S also contains all assignments a such that $[f(w)]; a$ appears in the projection of T onto the set $\{w, a\}$.

These restrictions ensure that each channel is used exactly once per cycle of the process. Since the i^{th} occurrence of any assignment can only depend on either the $(i-1)^{th}$ or i^{th} occurrence of a preceding assignment, all occurrence index offsets in the repetitive ER system that models such a process are either 0 or 1. An LR buffer can be initialized to either a state where the first communication action on the output channel precedes the first communication on the input channel or vice versa. In the former case, there is an initial communication on the output channel.

		Sink			
		$lo^k{\uparrow}$	$lo^k{\downarrow}$	$ro^k{\uparrow}$	$ro^k{\downarrow}$
Source	$li^j{\uparrow}$	σ_0^{jk}	σ_1^{jk}	ρ_0^{jk}	ρ_1^{jk}
	$li^j{\downarrow}$	σ_2^{jk}	σ_3^{jk}	ρ_2^{jk}	ρ_3^{jk}
	$ri^j{\uparrow}$	λ_0^{jk}	λ_1^{jk}	σ_4^{jk}	σ_5^{jk}
	$ri^j{\downarrow}$	λ_2^{jk}	λ_3^{jk}	σ_6^{jk}	σ_7^{jk}

TABLE I

CLASSIFICATION OF PATHS IN THE CONSTRAINT GRAPH OF A PROCESS

We state sufficient conditions for a system to have cycle time at most τ_0 in terms of the dynamic slack at τ_0 of each process in the system. We prove that these conditions are sufficient by considering all cycles in the collapsed constraint graph of the system. In order to facilitate this proof, we introduce the following notation for the paths in the collapsed constraint graph of a process P.

The *input variables* of a process P are defined to be variables that implement communication channels of P such that the variables are not assigned to by P. The *output variables* of a process P are defined to be the variables that implement communication channels of P and are assigned to by P. We classify the paths between transitions on input variables of P and output variables of P as follows. Each path has a label of the form a_n^{jk} where $a \in \{\sigma, \lambda, \rho\}$. Any path between a transition on a variable li^j and a transition on variable ro^k has $a = \rho$. Similarly, paths between a transition on variable ri^j and lo^k have $a = \lambda$. Paths between li^j and lo^k and those between ri^j and ro^k have $a = \sigma$. The superscript identifies the channels to which the source and sink of the path belong. The subscripts identify the input and output transitions, as shown in table I. We use the notation $\delta(p)$ to denote the sum of the delays of the rules that constitute the path p. Let $\epsilon(p)$ denote the sum of the occurrence index offsets along p. The notation $a_n^{jk}(P_i)$ is used to denote a a_n^{jk} path of process P_i.

Let the delays on paths between the input and output variables of a process, i be bounded by $f_i^{jk}, b_i^{jk}, w_i^{jk}, s_i^{jk}, x_i^{jk}$ and u_i^{jk} as in table II. The table also show the occurrence index offsets on these paths in three cases — no initial communication on the input or output channels(N), an initial communication on the output channel(S) and an initial communication on the input channel(R). If the occurrence index offset of a path is a greater than that shown in the table, the delay of the path is permitted to be $a\tau_0$ greater than the corresponding bound. Let l, m and n be processes such that channels (l, m) and (m, n) exist. Define $M((l, m), (m, n))$, the number of initial messages in process m between channels (l, m) and (m, n), to be $\frac{1}{2}$ if there is an initial communication on either channel (l, m) or (m, n), 0 otherwise. Let $d_{\min}((l, m), (m, n), \tau)$ be $\frac{f_m^{ln}}{\tau}$ and $d_{\max}((l, m), (m, n), \tau)$ be $\frac{\tau - 2b_m^{nl}}{2\tau}$.

First we consider systems comprised of processes with exactly one input channel and one output channel. We state restrictions on the collapsed constraint graphs of each process in the system which allow us to determine whether the system's cycle time is at or below the target τ_0.

Path(p)	$\delta(p)$	$\epsilon(p):N$	$\epsilon(p):S$	$\epsilon(p):R$	Path(p)	$\delta(p)$	$\epsilon(p):N$	$\epsilon(p):S$	$\epsilon(p):R$
$\lambda_0^{jk}(i)$	$\leq b_i^{jk}+\frac{\tau_0}{2}$	1	1	0	$\sigma_0^{jk}(i)$	$\leq x_i^{jk}$	1	1	0
$\lambda_1^{jk}(i)$	$\leq b_i^{jk}$	0	0	0	$\sigma_1^{jk}(i)$	$\leq u_i^{jk}+\frac{\tau_0}{2}$	1	1	1
$\lambda_2^{jk}(i)$	$\leq b_i^{jk}$	1	0	0	$\sigma_2^{jk}(i)$	$\leq x_i^{jk}+\frac{\tau_0}{2}$	1	1	1
$\lambda_3^{jk}(i)$	$\leq b_i^{jk}+\frac{\tau_0}{2}$	1	0	1	$\sigma_3^{jk}(i)$	$\leq u_i^{jk}$	0	0	1
$\rho_0^{jk}(i)$	$\leq f_i^{jk}$	0	1	0	$\sigma_4^{jk}(i)$	$\leq w_i^{jk}+\frac{\tau_0}{2}$	0	1	0
$\rho_1^{jk}(i)$	$\leq f_i^{jk}+\frac{\tau_0}{2}$	1	1	1	$\sigma_5^{jk}(i)$	$\leq s_i^{jk}$	0	0	0
$\rho_2^{jk}(i)$	$\leq f_i^{jk}+\frac{\tau_0}{2}$	0	1	1	$\sigma_6^{jk}(i)$	$\leq w_i^{jk}$	0	0	0
$\rho_3^{jk}(i)$	$\leq f_i^{jk}$	0	0	1	$\sigma_7^{jk}(i)$	$\leq s_i^{jk}+\frac{\tau_0}{2}$	1	0	1

TABLE II

BOUNDS ON DELAYS IN PROCESS i

Assumption 1: Let the input channel of any process i be labeled L^k and the output channel be labeled R^l. Furthermore, let the delays and occurrence index offsets of paths between the input and output variables of each process be bounded as in table II.

- For each process, i, $f_i^{kl}+b_i^{lk}\leq\frac{\tau_0}{2}$.
- For each process, i, inequalities (2)–(5) hold.
- Inequalities (6)–(9) hold for all pairs of processes i and j such that output channel R^l of j is the input channel L^k of i.

$$w_i^{kk}+x_i^{ll}\geq f_i^{lk}+b_i^{lk} \tag{2}$$
$$w_i^{kk}+u_i^{ll}\geq f_i^{lk}+b_i^{lk} \tag{3}$$
$$s_i^{kk}+x_i^{ll}\geq f_i^{lk}+b_i^{lk} \tag{4}$$
$$s_i^{kk}+u_i^{ll}\geq f_i^{lk}+b_i^{lk} \tag{5}$$

$$w_i^{kk}+x_j^{ll}\leq\frac{\tau_0}{2} \tag{6}$$
$$w_i^{kk}+u_j^{ll}\leq\frac{\tau_0}{2} \tag{7}$$
$$s_i^{kk}+x_j^{ll}\leq\frac{\tau_0}{2} \tag{8}$$
$$s_i^{kk}+u_j^{ll}\leq\frac{\tau_0}{2} \tag{9}$$

Next, we state sufficient conditions under which a system comprised of such processes has cycle time at most τ_0. This set of sufficient conditions is specified as a set of linear constraints with a subset of the variables being restricted to the integers. Note that since each process has exactly one input and one output channel, the process graph of such a system is a collection of independent rings and lines.

Lemma 1: Let $G=(V,E)$ be the process graph of a system, composed of processes that have exactly one input channel and one output channel, satisfying assumption 1. Let the variables $d_{(i,j),(j,k)}$ take on a value in the dynamic slack of process j between channels (i,j) and (j,k). Let the variables denoted S_{ab} be such that $S_{ab}+\sum_{i=0}^{|p|-2}M(e_i,e_{i+1})$ is in the dynamic slack of the pipeline corresponding to the path p, where p is a path with a as the first edge and b as the last edge. The system has cycle time at most τ_0 if there exist S_{ab} and $d_{a,b}$ such that equations (10)–(13) hold.

$$d_{a,b}\in[d_{\min}(a,b,\tau_0),d_{\max}(a,b,\tau_0)]$$
$$\forall a,b\in E:\mathrm{src}(b)=\mathrm{snk}(a) \tag{10}$$
$$S_{aa}=0\;\forall a\in E \tag{11}$$
$$S_{ab}=S_{ac}+d_{c,b}-M(c,b)+S_{bb}$$
$$\forall a,c,b\in E:c\in\mathcal{S}(a)\wedge\mathrm{snk}(c)=\mathrm{src}(b) \tag{12}$$
$$S_{ab}=-d_{b,a}+M(b,a)$$
$$\forall a,b\in E:b\in\mathcal{S}(a)\wedge\mathrm{snk}(b)=\mathrm{src}(a) \tag{13}$$

Proof: Intuitively, the lemma postulates that for a pipeline, p, composed of process with exactly one input channel and one output channel,

$$d_{\min}(p.in,p.out,\tau_0)\leq\sum_{i=0}^{|p|-2}d_{\min}(e_i,e_{i+1},\tau_0)$$

and

$$d_{\max}(p.in,p.out,\tau_0)\geq\sum_{i=0}^{|p|-2}d_{\max}(e_i,e_{i+1},\tau_0).$$

(10)–(13) enforce the requirement that it be possible for each ring and pipeline in the system to simultaneously contain a number of messages within its dynamic slack at τ_0.

Let C be the collapsed constraint graph of the system represented by G. We prove the lemma by showing for all cycles in C, the ratio of the delay along the cycle to the sum of occurrence index offsets along the cycle is at most τ_0.

Let a *handshake cycle* be a cycle in C such that all the vertices it traverses are either variables local to a process or are variables that implement exactly one channel. By enumerating all handshake cycles and computing upper bounds on the ratio of the delay to the sum of the occurrence index offsets along the cycle, it is easily seen that a handshake cycle cannot constrain the system's cycle time to be greater than τ_0.

Consider the collapsed constraint graph of a pipeline of $n>0$ processes $P=\{P_i\}$ such that the output channel of process P_i is an input channel of P_{i+1}.

Let the *critical path* between a pair of vertices in a collapsed constraint graph be the path, p, such that $\delta(p)-\tau_0\cdot\epsilon(p)$ is maximized. We will state the critical path in a pipeline of processes that satisfy the claim. We use these critical paths to sh no cycle in the collapsed constraint graph of a ring of processes satisfying the claim can constrain the ring's cycle time to be greater than τ_0.

Consider all paths p such that $\mathrm{src}(p) \in \{P_0.li\uparrow, P_0.li\downarrow\}$, $\mathrm{snk}(p) \in \{P_{n-1}.ro\uparrow, P_{n-1}.ro\downarrow\}$ and p only traverses $\rho(P_i)$ paths. Let there be no initial messages on the input channel of P_0 and the output channel of P_{n-1}. It can be shown that if there are m messages on the pipeline, the critical $\rho_0(P), \rho_1(P), \rho_2(P)$ and $\rho_3(P)$ paths have delay at most $\sum f_i$, $\frac{\tau_0}{2} + \sum f_i$, $\frac{\tau_0}{2} + \sum f_i$ and $\sum f_i$ respectively. The sum of occurrence index offsets is respectively m, $m+1$, m and m.

Similarly, consider paths p such that $\mathrm{src}(p) \in \{P_{n-1}.ri\uparrow, P_{n-1}.ri\downarrow\}$, $\mathrm{snk}(p) \in \{P_0.lo\uparrow, P_0.lo\downarrow\}$ and p only traverses $\lambda(P_i)$ paths. It can be shown that when the number of processes is n such that $n = 2x + y$, where x and y are non-negative integers such that $y \in \{0, 1\}$, the critical $\lambda_0(P), \lambda_1(P), \lambda_2(P)$ and $\lambda_3(P)$ paths have delays at most $y \cdot \frac{\tau_0}{2} + \sum b_i$, $(1-y) \cdot \frac{\tau_0}{2} + \sum b_i$, $(1-y) \cdot \frac{\tau_0}{2} + \sum b_i$, and $y \cdot \frac{\tau_0}{2} + \sum b_i$ respectively. The sum of the occurrence index offsets is respectively $x + y - m$, $x - m$, $x - m + 1$ and $x + y - m$.

Consider a ring, r, of n processes $\{P_i\}$ such that the output channel of process P_i is the input channel of process P_{i+1} and the output channel of P_{n-1} is the input channel of P_0. Let P_r be the pipeline $P_r = \{P_i\}$. Any cycle in the collapsed constraint graph of r that traverses only ρ paths in each P_i must fall into one of three cases.

1) The cycle consists of a ρ_0 path in P_r: (10)–(13) guarantee that $\sum d_{e_i, e_{(i+1) \bmod n}} = \sum M(e_i, e_{(i+1) \bmod n})$ for all rings. Thus, $\sum \frac{f_i}{\tau_0} \leq m$. Recall that the critical ρ_0 path has delay at most $\sum f_i$ and the sum of the occurrence index offsets along this path is m. Thus the cycle cannot constrain the cycle time to be greater than τ_0.
2) The cycle consists of a ρ_3 path in P_r: The same analysis as for the previous case holds.
3) The cycle consists of a ρ_1 path in P_r followed by a ρ_2 path in P_r: The cycle has delay $\tau_0 + \sum 2f_i$ and the sum of the occurrence index offsets is $2m+1$. Since (10)–(13) imply that $\sum \frac{f_i}{\tau_0} \leq m$, such a cycle cannot constrain the cycle time to be greater than τ_0.

A similar analysis can be used to show that no cycle in the collapsed constraint graph of ring $\{P_i\}$ consisting solely of λ paths can constrain the cycle time to be greater than τ_0.

An inductive argument can then be used to show that no cycle in C constrains the cycle time to be greater than τ_0, proving the lemma. ■

Next we consider systems composed of processes that have at least one input channel and at least one output channel.

Assumption 2:
- Assumption 1 is satisfied for any system obtained by projecting each process onto a pair of input channel and an output channel.
- If L^k and L^n are input channels of process i and R^l and R^m are output channels of i, then $f_i^{kl} + b_i^{mn} \leq \frac{\tau_0}{2}$.
- For any pair of output channels R^j and R^k and any input channel L^i of process m, the constraints in table III hold.
- For any pair of input channels L^j and L^k and any output channel R^l of process m, the constraints in table IV hold.

Path(p)	$\delta(p)$	$\epsilon(p)$
σ_4^{jk}	$\leq \delta(\sigma_4^{kk})$	$\epsilon(\sigma_4^{kk})$
σ_5^{jk}	$\leq \delta(\sigma_5^{kk})$	$\epsilon(\sigma_5^{kk})$
σ_6^{jk}	$\leq \delta(\sigma_6^{kk})$	$\epsilon(\sigma_6^{kk})$
σ_7^{jk}	$\leq \delta(\sigma_7^{kk})$	$\epsilon(\sigma_7^{kk})$

(a)

Path	Delay
σ_4^{jk}	$\leq f_m^{ik} + \frac{\tau_0}{2}$
σ_5^{jk}	$\leq f_m^{ik}$
σ_6^{jk}	$\leq f_m^{ik}$
σ_7^{jk}	$\leq f_m^{ik} + \frac{\tau_0}{2}$

(b)

TABLE III

CONSTRAINTS ON DELAYS OF PATHS IN PROCESSES WITH MULTIPLE OUTPUTS

Path(p)	$\delta(p)$	$\epsilon(p)$
σ_0^{jk}	$\leq \delta(\sigma_0^{jj})$	$\epsilon(\sigma_0^{jj})$
σ_1^{jk}	$\leq \delta(\sigma_1^{jj})$	$\epsilon(\sigma_0^{jj})$
σ_2^{jk}	$\leq \delta(\sigma_2^{jj})$	$\epsilon(\sigma_0^{jj})$
σ_3^{jk}	$\leq \delta(\sigma_3^{jj})$	$\epsilon(\sigma_0^{jj})$

(a)

Path	Delay
σ_0^{jk}	$\leq b_m^{lk}$
σ_1^{jk}	$\leq b_m^{lk} + \frac{\tau_0}{2}$
σ_2^{jk}	$\leq b_m^{lk} + \frac{\tau_0}{2}$
σ_3^{jk}	$\leq b_m^{lk}$

(b)

TABLE IV

CONSTRAINTS ON DELAYS OF PATHS IN PROCESSES WITH MULTIPLE INPUTS

When the processes have greater than one input channel and greater than one output channel, two distinct pipelines may have the same input channel and output channel. Thus, the difference in the number of messages in the two pipelines is constant and equal to the difference in the number of initial messages in the pipelines. Intuitively, (10)–(13) ensure that in such cases, it is possible for both pipelines to simultaneously contain a number of messages within their dynamic slack.

Recall that if a message is sent on channel (a, b) then a message is inserted into all pipelines that have input channel (a, b). Similarly if a message is received on channel (c, d) then a message has been removed from all pipelines that have (c, d) as their output channel. Constraints of the form (14)–(17) ensure that it is possible for all pipelines to simultaneously contain a number of messages within their dynamic slack whilst still obeying this relationship.

$$
\begin{aligned}
S_{ab} &= K_{ac} + d_{i,c} - M(i,c) + M(i,b) + S_{bb} \\
&\qquad \forall a,b,i,c \in E : c \in \mathcal{K}(a), \mathrm{src}(c) = \mathrm{snk}(i) = \mathrm{src}(b) \quad (14) \\
K_{ab} &= K_{ac} - d_{b,c} - M(b,c) - S_{bb} \\
&\qquad \forall a,b,c \in E : c \in \mathcal{K}(a), \mathrm{src}(c) = \mathrm{snk}(b) \quad (15) \\
K_{ab} &= S_{ac} - d_{c,o} + M(c,o) - M(b,o) - S_{bb} \\
&\qquad \forall a,b,o,c \in E : c \in \mathcal{S}(a), \mathrm{snk}(c) = \mathrm{src}(o) = \mathrm{snk}(b) \quad (16) \\
K_{ab} &= -d_{i,a} + M_{i,a} - M_{i,b} \\
&\qquad \forall a,b,i \in E : b \in \mathcal{K}(a), \mathrm{src}(a) = \mathrm{src}(b) = \mathrm{snk}(i) \quad (17)
\end{aligned}
$$

Lemma 2: Let $G = (V, E)$ be the process graph of a system, composed of processes that have at least one input channel and at least one output channel, satisfying assumption 2. T

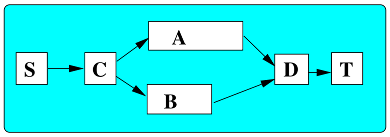

Fig. 5. Pipeline with re-convergent fanout

tem has cycle time at most τ_0 if there exist S_{ab}, $d_{a,b}$ and K_{ab} such that equations (10)–(17) hold.

Proof: We now proceed to show that no cycle in the system's collapsed constraint graph constrains the cycle time to be greater than τ_0. Consider a pipeline with reconvergent fanout, as in figure 5. The critical ρ_0 path in pipeline A has delay at most $\sum f_i$ and the sum of occurrence index offsets is m_A, the number of initial messages in pipeline A. Similarly, the critical λ_0 path in B has delay at most $y_B \cdot \frac{\tau_0}{2} + \sum b_i$ and the sum of occurrence index offsets is $x_B + y_B - m_B$ where $n_B = 2x_B + y_B$ is the number of processes in pipeline B and m_B the number of initial messages in the pipeline. Let the cycle contain a σ_4^{jk} path in process C and a σ_0^{jk} path in D.

The delay along this cycle is bounded by

$$\sum_{i\in A} f_i + \sum_{i\in B} b_i + f_C + b_D + (1 + y_B)\frac{\tau_0}{2} \tag{18}$$

The sum of the occurrence index offsets along this path is given by $m_A + x_B + y_B - m_B + 1$. Assumption 2 and (10)–(17) guarantee that there exists $K_{(C,A)(C,B)}$ such that:

$$K_{(C,A)(C,B)} \geq \sum_{i\in A} \frac{f_i}{\tau_0} + \frac{f_C}{\tau_0} - m_A \tag{19}$$

$$K_{(C,A)(C,B)} \leq \sum_{i\in B} \frac{\tau_0 - 2b_i}{2\tau_0} + \frac{\tau_0 - 2b_D}{2\tau_0} - m_B \tag{20}$$

From (19) and (20) it is easily seen that such a cycle cannot constrain the cycle time of the system to be greater than τ_0. A similar analysis can be used to eliminate all other cycles in the case of reconvergent fanout.

An inductive argument can then be used to show that any cycle in the system's constraint graph does not restrict the system's cycle time to be greater than τ_0. ■

IV. Slack Matching Algorithm

In this section, we describe an algorithm for slack matching a system composed of processes satisfying assumption 2. We formulate the slack matching problem as a mixed integer linear program (MILP) that can be solved to determine the placement of slack matching buffers. Channels that carry data are modeled as data-less channels. If the system is not closed, a source is connected to each of the system's input channels and a sink to each of the system's output channels. It is assumed that the environment is such that it does not constrain the system's cycle time to be greater than the target, τ_0. We state the slack matching problem as a MILP and show how to generate this MILP efficiently.

Consider a process graph $G = (V, E)$ representing a system comprised of processes satisfying assumption 2. The system represented by G is slack matched at τ_0 if lemma 2 holds.

A. MILP for slack matching

Slack matching is performed by adding buffers along communication channels in such a manner that the resulting system satisfies constraints (10)–(17). Slack matching only changes a system by adding buffers to communication channels. Let b be the buffer that is used for slack matching a system. Let $b.L$ and $b.R$ respectively be the input and output channels of b. $[d_{\min}(b.L, b.R, \tau_0), d_{\max}(b.L, b.R, \tau_0)]$ denotes the dynamic slack of b. Replacing constraints of the form $S_{ee} = 0$ by ones of the form (21) and (22), we obtain the set of constraints from lemma 2 for the system with n_e slack matching buffers added to each channel e. Slack matching a system now reduces to determining non-negative integers n_e such that the system of equations (10),(12)–(17) and (21)–(22) is satisfied.

$$S_{ee} \in [n_e \cdot d_{\min}(b.L, b.R, \tau_0), n_e \cdot d_{\max}(b.L, b.R, \tau_0)] \quad \forall e \in E \tag{21}$$

$$n_e \in \mathbb{N} \forall e \in E \tag{22}$$

Since there may be multiple solutions to the set of equations, any cost function linear in n_e may be used to drive the optimization.

B. Generating the MILP

The set of constraints for slack matching a system can be computed in $O(m^2n^2)$ time where $m = |E|$ and $n = |V|$.

Constraints of the form (21) and (22) can be generated in $O(m)$ time by looping over the edge set.

It takes $O(mn+m^2)$ time to generate $m \times m$ matrices $\mathcal{S}$ and $\mathcal{K}$, such that

$$\mathcal{S}_{ij} = \begin{cases} 1, & i \in \mathcal{S}(j) \\ 0, & i \notin \mathcal{S}(j) \end{cases} \tag{23}$$

and

$$\mathcal{K}_{ij} = \begin{cases} 1, & i \in \mathcal{K}(j) \\ 0, & i \notin \mathcal{K}(j) \end{cases} \tag{24}$$

This is done by running m breadth-first searches(BFS) on G_m, the undirected version of the graph obtained by removing edge m from G. The *undirected version* of a directed graph, $G = (V, E)$ is the graph $G' = (V, E')$ where E' is such that $\forall (u, v) \in E : (u, v) \in E' \wedge (v, u) \in E'$. The BFS on G_m is rooted at $\mathrm{snk}(m)$ and is modified so that for each vertex v, it records all the edges in G_m that could possibly be the last edge on a path from $\mathrm{snk}(m)$ to v. The matrices $\mathcal{S}$ and $\mathcal{K}$ can be constructed from these lists of edges.

Given matrix $\mathcal{S}$, constraints of the form (10) and (13) can be generated in $O(m^2)$ time by looping over all pairs of edges.

There are $O(m^2n)$ 3-tuples of edges (a, b, c) such that $c \in \mathcal{S}(a) \wedge \mathrm{snk}(c) = \mathrm{src}(b)$. Given matrices $\mathcal{S}$ and $\mathcal{K}$, constraints of the form (12),(15) and (17) can be generated by simply looping over all such 4-tuples.

There are $O(m^2n^2)$ 4-tuples of edges (a, b, c, i) such that $c \in \mathcal{K}(a) \wedge \mathrm{src}(c) = \mathrm{snk}(i) = \mathrm{src}(b)$. Given matrices $\mathcal{K}$ and $\mathcal{S}$, constraints of the form (14) and (16) can be generated in $O(m^2n^2)$ time by looping over all such 4-tuples.

Channel	# Buffers (hand)	# Buffers (MILP)
ExtControl - CBUFE	1	1
ExtControl - Router	1	1
ExtControl - IntCtrl	1	1
CBUF - IntCtrl	1	1
Router - SplD1	2	2
Router - MrgI0	1	1
Router - MrgExt	3	3
PCNH - PCIH	1	0
PCNH - PCUH	1	1
PCNL - PCUL	1	1
PCIL - PCUL	1	1

TABLE V

SLACK MATCHING BUFFERS FOR LUTONIUM FETCH

Owing to cycle time constraints on the internal cycle, the number of channels of a process is usually bounded, and significantly smaller than n. In this case, the constraints can be generated in $O(k^4 n^2)$ time, where k is the bound on the number of channels of a process.

V. RESULTS

The algorithm from section IV was implemented in Modula-3 [8] and used with `glpsol`, a freely available MILP solver. Two large examples were studied, the fetch loop of the Lutonium [7], an asynchronous 8051 microcontroller and a control loop in the fetch unit of the MiniMIPS microprocessor [6].

A. Example I: Lutonium Fetch Loop

This algorithm was used to slack match the fetch loop of the Lutonium microcontroller. The objective function minimized was the estimated energy consumption of the slack matching buffers. Whilst the instruction memory is not implemented as a pipeline of half buffers, it can be modeled as one. The memory is modeled as a stage such that $d_{\max} = d_{\min}$, with $d_{\min}$ being determined by the memory's latency.

Figure 6 shows the fetch loop of the Lutonium microcontroller.

Table V shows the buffers needed to slack match the system. The table also lists the results of slack matching when performed by hand on the system. Observe that there are fewer buffers on the byte channel in the pc increment loop when slack matching is performed using this algorithm, all other channels have an identical number of buffers.

B. Example II: Control Loop of MiniMIPS

Figure 7 shows a loop in the fetch of the MiniMIPS. Table VI shows the buffers required to slack match this loop. It also shows the results when slack matching was performed by hand. Note that extra buffers included when slack matching was performed by hand may be needed because a ring composed of a mixture of half buffers and full buffers was not included when generating the MILP.

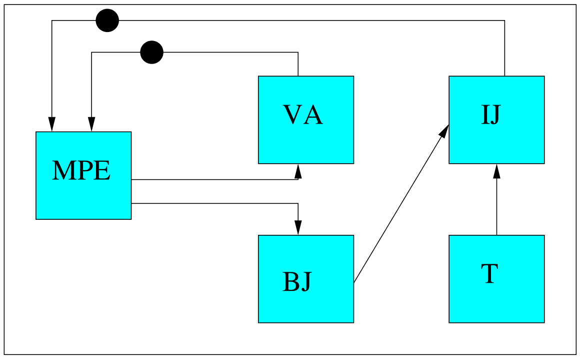

Fig. 7. Control Loop in the MiniMIPS fetch

Channel	# buffers (hand)	# buffers (MILP)
VA-MPE	4	1
IJ-MPE	4	1

TABLE VI

SLACK MATCHING BUFFERS FOR THE CONTROL LOOP IN THE MINIMIPS FETCH

VI. ALGORITHM FOR SLACK MATCHING CIRCUITS COMPOSED OF FULL-BUFFERS AND HALF-BUFFERS

We have presented an algorithm for slack matching systems composed of solely of half-buffers. However, many systems of interest are composed of processes with heterogenous static slacks. In this section, we present an algorithm to slack match such systems. We require that the buffers used for slack matching, referred to as slack matching buffers, satisfy assumption 1. This technique does not rely on the abstraction of dynamic slack, instead the algorithm determines a number of buffers of to be added to each channel such that there are no cycles in the resulting system's collapsed constraint graph that constrain the cycle time to be greater than τ_0.

We restrict our attention to collections of HSEs composed of processes that can be written as a straight-line handshaking expansion [1] of the form $P \equiv S; *[T]$ such that no assignment appears more than once in T. Given the collapsed constraint graph, C, of the system, S, to be slack matched, consider the collapsed constraint graph C' of the system S' obtained by adding n_{uv} slack matching buffers to each channel (u, v) in S. A set of linear constraints can be derived to determine the values of n_{uv} such that S' is slack matched. Thus, if an objective function linear in n_{uv} is used, S can be slack matched by solving an MILP.

Recall from Burns [1] that the cycle time of a repetitive ER system, (E, R), can be determined as the minimum τ for which there exist offsets $a_i \geq 0$ such that

$$a_v - a_u + \epsilon_{uv}\tau \geq \alpha_{uv} \quad \forall (u, v, \alpha_{uv}, \epsilon_{uv}) \in R. \qquad (25)$$

The number of edges and vertices in the collapsed constraint graph of a pipeline, P_{uv}, of n_{uv} slack matching buffers depends on n_{uv}. Each of these edges has a constant delay and occurrence index offset. We will show how to represent the collapsed constraint graph of P_{uv} as a graph with a c

0-7695-2498-2/06 $20.00 © 2006 IEEE

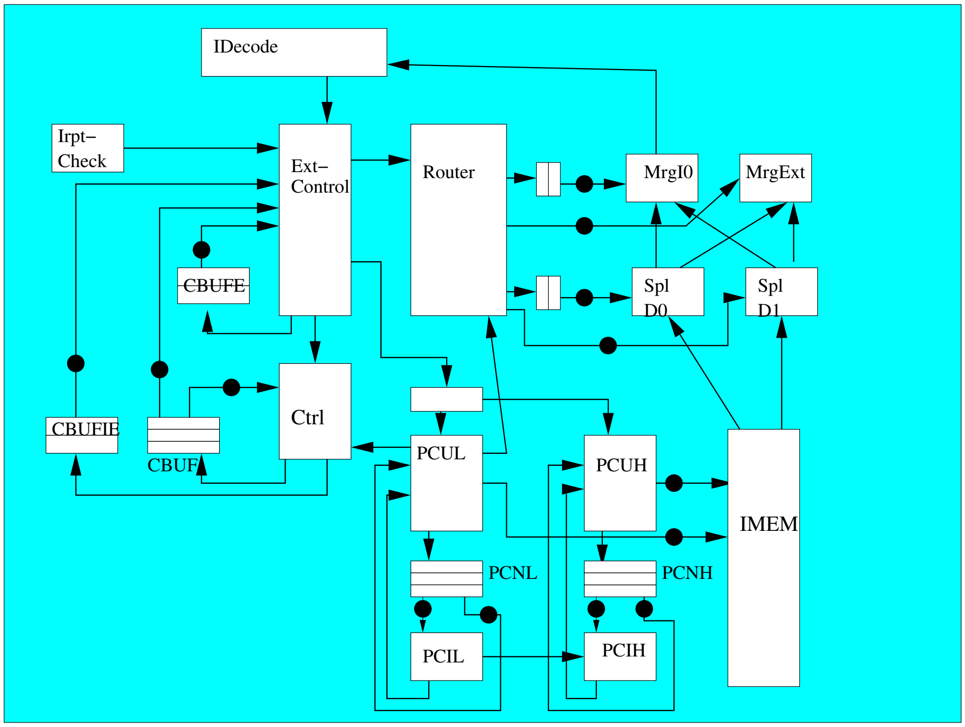

Fig. 6. Lutonium fetch loop

number of edges and vertices such that the occurrence index offsets and delays of the edges depend on n_{uv}. Let x_{uv} and y_{uv} be non-negative integers such that $n_{uv} = 2x_{uv} + y_{uv}$ with $y_{uv} \leq 1$. We represent the collapsed constraint graph of P_{uv} as shown in figure 8. The vertices $li\uparrow$ and $ro\uparrow$ are duplicated for the sake of clarity.

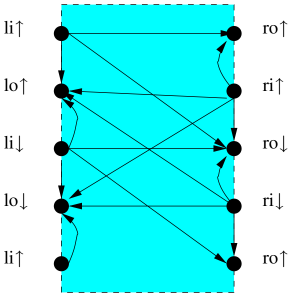

Fig. 8. Constraint graph of P_{uv}

Recall that the slack matching buffers satisfy assumption 1, thus as shown in the proof of lemma 1, when $n_{uv} \geq 1$, the delays along critical $\lambda(P_{uv})$ and $\rho(P_{uv})$ paths can be expressed as linear functions of x_{uv} and y_{uv}. The sum of the occurrence index offsets along these paths can also be expressed as linear functions of x_{uv} and y_{uv}. Similar techniques to those used in the proof of lemma 1 can be used to determine the critical $\sigma(P_{uv})$ paths and show that the delay and sum of occurrence index offsets along these paths are constant for $n_{uv} \geq 1$.

The delay of any path between a transition on an input variable of P_{uv} and a transition of an output variable of P_{uv} is zero when $n_{uv} = 0$. When $n_{uv} = 0$, the representation of P_{uv} can introduce cycles in C' that would not be present, were the pipeline P_{uv} removed and the corresponding input and output variables of P_{uv} connected with wires. In particular, when $n_{uv} = 0$, $\lambda_1(P_{uv}), \lambda_2(P_{uv}), \rho_1(P_{uv}), \rho_2(P_{uv})$ and $\sigma_*(P_{uv})$ paths should not exist in the collapsed constraint graph of P_{uv}. Thus care must be taken to ensure that the introduction of these paths does not result in cycles that artificially constrain the cycle time of S' to be greater than τ_0.

Let P_{rs} be a pipeline of slack matching buffers and t be a transition on an input variable of P_{rs}. Let A_t be the critical path between $P_{uv}.ro\uparrow$ and t and B_t be the critical path between $P_{uv}.ro\downarrow$ and t such that A_t and B_t do not traverse edges in the constraint graph of any pipeline of slack matching buffers. When the delay of the $\rho_1(P_{uv})$ edge is set to zero, and its occurrence index offset is chosen to be greater than (26), there exists a cycle in C' that traverses $\rho_1(P_{uv})$ which constrains the cycle time to be greater than τ_0 only if there exists a cycle in C' that traverses $\rho_0(P_{uv})$(but not $\rho_1(P_{uv})$) and constrains the cycle time to be greater than τ_0.

$$\max_t \left\lceil \frac{\delta(B_t) - \delta(A_t)}{\tau_0} \right\rceil + \epsilon(A_t) - \epsilon(B_t). \qquad (26)$$

Similar lower bounds on the occurrence index offsets of $\rho_2(P_{uv})$, $\lambda_1(P_{uv})$, and $\lambda_1(P_{uv})$ paths can be derived. In order to determine a lower bound on the occurrence index offset of the $\sigma_0(P_{uv})$ path, consider any path, p, from $P_{uv}.ro\uparrow$ $P_{uv}.ri\uparrow$ that does not traverse any edges in the collapsed constraint graph of any pipeline of slack matching buffers. The occurrence index offset of the $\sigma_0(P_{uv})$ path must be at least $\epsilon(p)$. The occurrence index offsets of the other $\sigma_i(P_{uv})$ edges can be determined in a similar manner.

In order to express the delays and occurrence index offsets of the various paths in P_{uv} as a linear function of the program variables, it is useful to determine whether n_{uv} is zero. We do this by introducing a binary variable z_{uv} such that $z_{uv} \leq n_{uv} \leq z_{uv} \cdot N_{\max}$ where $N_{\max}$ is the maximum number of slack matching buffers that can be placed on any channel.

Thus, given C', we can construct inequalities of the form (25). Note that τ is set to be τ_0, and ϵ_{ab} and α_{ab} may be linear functions of the three variables x_{uv}, y_{uv} and z_{uv} if the rule (a, b) corresponds to a path in the collapsed constraint graph of P_{uv}. Furthermore, we need to introduce the constraints $n_{uv} = 2x_{uv} + y_{uv}$ and $z_{uv} \leq n_{uv} \leq z_{uv} \cdot N_{\max}$. We can then solve a MILP with these constraints to determine n_{uv} such that the objective function is minimized. The objective function must be linear in n_{uv}.

VII. Prior Work

Chapter 2 of Lines [4] studies asynchronous pipelines and presents some necessary conditions for a system composed of half-buffers to be slack matched. Pénzes [9] presents an execution model for production rule sets. Using this model, the slack matching problem is formulated as an integer linear programming problem for some examples. However, the integer linear program needed to slack match any given system is not specified. No comment is made on the size of the integer linear program generated relative to the size of the system being slack matched. This method is similar to the technique for slack matching presented in section VI. We explicitly state the integer linear program that needs to be solved in order to slack match a system. Chapter 5 of Wong [11] presents a method for slack matching systems composed of processes with the PCHB reshuffling. If the delays of all processes are identical, the slack matching algorithm presented is similar to that in section IV. However, for heterogeneous systems Wong [11] needs to study all the paths in the FBI graph of a system in order to set up the slack matching problem. FBI graphs are an execution model for systems composed of processes with the PCHB reshuffling. The FBI graph of process is essentially a simplified collapsed constraint graph of the process In contrast, we present a method to frame the slack matching problem as a mixed integer linear program based on the process graph even in the heterogenous case. We show how to construct this program in polynomial time.

VIII. Conclusions and Future Work

Two methods of expressing the slack matching problem as a MILP have been presented.

The method in section IV provides a set of conditions for slack matching systems composed of a specified class of half-buffers. A polynomial time algorithm has been presented to generate the MILP. This method of generating an MILP, and then solving using general purpose MILP solvers has been applied to circuits from the Lutonium [7] and the MiniMIPS [6]. Similar conditions to those in section III can be derived for slack matching circuits comprised of a restricted set of full-buffers. The next step would be to derive similar constraints for systems comprised of both full-buffers and half-buffers.

For the circuits studied so far, solving the MILP has not taken large amounts of time. However, if solving the MILP for slack matching larger systems does take excessive amounts of time, it faster heuristics should be explored.

The method in section VI imposes fewer restrictions on the processes that comprise the system being slack matched; however the MILP generated is larger than that generated using the first method. The next step is to extend this method to permit processes with conditional communication.

Acknowledgments

The research described in this paper was supported by the Defense Advanced Research Projects Agency (DARPA) as part of the program in Power-Aware Computing and Communications (PAC/C) and monitored by the Air Force Office of Scientific Research.

References

[1] S.M. Burns. *Performance Analysis and Optimization of Asynchronous Circuits*. PhD thesis, California Institute of Technology, 1990.

[2] S. Kim and P. Beerel. Pipeline optimization for asynchronous circuits:complexity analysis and an efficient optimal algorithm. In *Proc. International Conference on Computer-Aided Design*, 2000.

[3] C. Leiserson, F. Rose, and J. Saxe. Optimizing synchronous circuitry by retiming. In *Third Caltech Conference On VLSI*, March 1993.

[4] A.M. Lines. Pipelined asynchronous circuits. Master's thesis, California Institute of Technology, 1995.

[5] R. Manohar and A.J. Martin. Slack elasticity in concurrent computing. In J. Jeuring, editor, *Proc. 4th International Conference on the Mathematics of Program Construction*, Lecture Notes in Computer Science 1422, pages 272–285. Springer Verlag, 1998.

[6] A.J. Martin, A. Lines, R. Manohar, M. Nyström, P. Pénzes, R. Southworth, U. Cummings, and T.K. Lee. The design of an asynchronous MIPS R3000 microprocessor. In *Proc. 17th Conference on Advanced Research in VLSI*, 1997.

[7] A.J. Martin, M. Nyström, K. Papadantonakis, P.I. Pénzes, P. Prakash, C.G. Wong, J. Chang, K.S. Ko, B. Lee, E. Ou, J. Pugh, E. Talvala, J.T. Tong, and A Tura. The Lutonium: A sub-nanojoule asynchronous 8051 microcontroller. In *Proc. 9th IEEE Intl Symposium on Advanced Research in Asynchronous Circuits and Systems*, May 2003.

[8] G. Nelson. *Systems programming with Modula-3*. Prentice Hall, 1991.

[9] P. Pénzes. Pipeline composition for asynchronous circuits. unpublished, September 1999.

[10] T. Williams. Latency and throughput tradeoffs in self-timed asynchronous pipelines and rings. Technical report, Stanford, 1990.

[11] C.G. Wong. *High-Level Synthesis and Rapid Prototyping of Asynchronous VLSI Systems*. PhD thesis, California Institute of Technology, 2004.

Appendix

All variables in a collection of HSE are boolean variables. The statements in a HSE are assignments to these variables. $a\uparrow$ and $a\downarrow$ respectively denote the assignment of the values **true** and **false** to variable a. Sequential composition is denoted by ; and , denotes parallel composition. The deterministic selection statement $[G_1 \rightarrow S_1 \,[]\, G_2 \rightarrow S_2]$ waits for one of the guards G_i to be true and then executes S_i. The looping statement $*[G \rightarrow S]$ executes S while the guard G holds. The statement $[G \rightarrow \textbf{skip}]$ is abbreviated $[G]$. Similarly, the loop $*[\texttt{true} \rightarrow S]$ is abbreviated $*[S]$.