Throughput Optimization of Quasi Delay Insensitive Circuits via Slack Matching

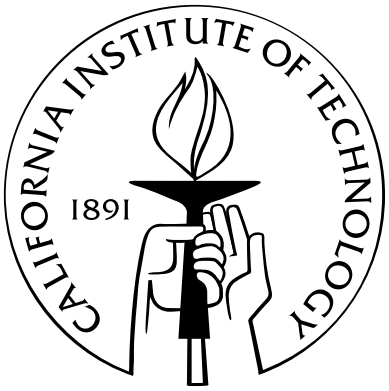
Thesis by

Piyush Prakash

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2008

(Submitted December 2007)

To my parents Dr. Shiv and Mrs. Neelam Prakash.

Acknowledgements

I would like to begin by thanking my advisor, Prof. Alain J. Martin for his patience, guidance and encouragement over the past years. He introduced me to the very interesting field of designing circuits without clocks. His search for simple and elegant solutions will be a guiding force in my continued learning.

I would also like to thank Prof. André DeHon for teaching me a very interesting class in electronic design automation, and for his detailed feedback on early drafts of my thesis. I thank the other members of my committee, Prof. K. Mani Chandy and Prof. Chris Umans for their constructive feedback on my thesis.

Another very important part of my life as a graduate student has been the other members of the Asynchronous VLSI research group at Caltech. I would like to thank Mika Nyström, Paul Pénzes, Catherine Wong, Karl Papadantonakis, Wonjin Jang, Jonathan Dama, Weiwei Yang, Chris Moore and Sean Keller. In particular I would like to thank them for listening, commenting and debating the research presented here at our Friday morning group meetings. I would also like to thank the departmental assistants Jeri Chittum, Betta Dawson and Diane Goodfellow for helping me navigate through the adminstrative parts of life as a graduate student.

My life as a graduate student has been enriched by some close friendships I have formed at Caltech. I would like to thank Matthew Mattson, Frosso Seitaridou and Scott Miserendino for their friendship over the years. They have been instrumental in keeping me sane over the years, making sure that I took breaks from work, encouraging me when I was down and being great friends. I thank Richard Korn for helping me keep sane by reminding me to exercise and encouraging me to train for and run in a marathon and half marathon. The training runs were a great way to clear my mind.

It was on these runs that some of the ideas in this thesis were formed. I would also like to thank my brother Punit Prakash, for listening to me when I was down, providing encouragement and an outside perspective.

Lastly, I would like to thank my parents Dr. Shiv and Mrs. Neelam Prakash for their constant support, encouragement of and commitment to my education. Words cannot convey the immense depth of my gratitude to them. I dedicate this thesis to them.

Abstract

Though the logical correctness of an asynchronous circuit is independent of implementation delays, the cycle time of an asynchronous circuit is of great importance to the designer. Oftentimes, the insertion of buffers to such circuits reduces the cycle time of the circuit without affecting the logical correctness of the circuit. This optimization is called slack matching. In this thesis the slack matching problem is formulated. I show that this problem is NP-complete via a reduction from subset sum. I describe two methods for expressing slack matching as a mixed integer linear program(MILP). The first method is applicable to any QDI circuit, while the second method produces a smaller MILP for circuits comprised solely of half buffers. These two formulations of slack matching were applied to the design of a fetch loop in an asynchronous micro-controller. Slack matching reduced the cycle time of the circuit by a factor of 3. For a circuit composed of 14 byte wide processes and a 8k instruction memory, 30s were required to generate the first MILP. It was solved in 2s. When the memory is modeled as a pipeline of half buffers, the second MILP could be formulated in 0.1s and solved in 0.6s. This MILP had half the number of integer variables as the first formulation.

Contents

List of Figures

Chapter 1

Introduction

Asynchronous circuits use local handshakes instead of a global clock for synchronization. Many asynchronous circuits are designed as collections of processes that share information only via message passing. The *cycle time* of a circuit is the mean time between successive occurrences of the same transition in the circuit. It has been observed that the insertion of additional processes that act as buffers can greatly reduce a circuits' cycle time [26, 12]. Slack matching is the process of inserting additional processes in order to reduce its cycle time.

In this thesis, two methods of formulating the slack matching problem as a mixed integer linear program(MILP) are presented. The first MILP is applicable only to half buffers. The second MILP, is applicable to a more general class of circuits. However, the second MILP requires more integer variables. During the design of the Lutonium [19] micro-controller, certain decisions were influenced by the anticipated need for slack matching buffers on certain channels. That is, the designer expected slack matching buffers to be inserted on certain channels. Therefore, rather than attempting to fit certain computations in one process, the computation was distributed over two or more processes. Even for such a circuit, slack matching reduced the cycle time by a factor of 3.

This Chapter is organized as follows. First, a simple example to motivate the slack matching problem is described in Section 1.1. Second, in Section 1.2 some background information about asynchronous circuits is provided. Next, the slack matching problem is defined in Section 1.3. In Section 1.4 a more detailed example of

the slack matching problem is described. The prior work on this problem is described in Section 1.5. The contributions of this thesis are summarized in Section 1.6. Finally in Section 1.7, an outline of the remainder of this thesis is provided.

1.1 Motivating Example

In this section, an example to motivate the need for slack matching is described. This section begins with a description of the properties of two processes with a channel between them. Next, lower bounds on the cycle time of a circuit are computed. It is shown how these lower bounds change when an additional process is added to the circuit.

Recall that instead of a global clock, asynchronous circuits use a local handshake for synchronization. Consider two processes, P and Q, with a communication channel from P to Q, that is the channel permits P to send data to Q. There is a set of variables assigned to by P that encode the data to be sent on the channel. Similarly, there is a variable assigned to exclusively by Q, that acknowledges the data. This variable signals to P that Q has observed that data and that the data may be changed. The communication actions are blocking, that is P cannot begin the next communication on this channel until it has received an acknowledgment from Q.

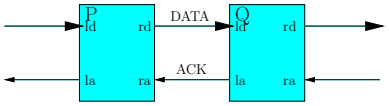


Figure 1.1: Two processes with a channel between them.

Consider a ring of processes, each having one input channel and one output channel. Let each process on this ring repeatedly wait for one message on its input channel. When a message arrives, the process produces a message on its output and acknowledges a message on its input. The next message on the input channel cannot arrive until this acknowledgment has been produced. Similarly, the process cannot send a subsequent message on its output channel until the previous message sent has been acknowledged. Figure 1.2 shows an example of such a ring. Let one process on this

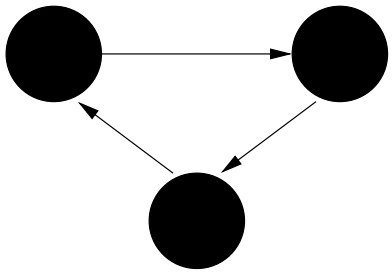


Figure 1.2: Ring of processes.

ring begin by sending a message on its output channel. Let all other processes begin by receiving a message and forwarding the message on their output channels. There is said to be one message on this ring. Let f, be the delay of each process between receiving a message on its input and sending a message on its output. One lower bound on the cycle time, τ, of a ring with N processes and m messages is:

$$\tau \geq \tau_f = \frac{Nf}{m}.$$

Since each process is of finite size, it can contain only a finite number of messages. Thus, each process can acknowledge only a finite number of messages on its input channel before having to wait for an acknowledgment on its input channel. For some processes, a pipeline of N processes can contain at most $\frac{N}{2}$ messages. Thus, two such processes are required to contain one message. Let the processes be such that a ring of N processes can contain at most $\frac{N}{2}$ messages. The number of additional messages the ring can contain, called the number of holes, is given by $\frac{N}{2} - m$ if the ring contains m messages initially. Let b be the delay between each process receiving an acknowledgment on its output channel and the processes sending an acknowledgment on its input channel. Another lower bound on the cycle time, τ, of a ring with N processes and h holes is:

$$\tau \geq \tau_b = \frac{Nb}{h}.$$

If the processes are all half buffers, $h = \frac{N}{2} - m$, thus a lower bound on the cycle is:

$$\tau \geq \tau_b = \frac{Nb}{\frac{N}{2} - m}.$$

Furthermore, the cycle must be at least as large as the time it takes for each process to receive on its input channel, send on its output channel and return to the state where it is ready to receive on its input channel. This is called the internal cycle time of the process.

Thus, for a ring of processes that have identical internal cycle time, τ_i, the cycle time, τ, of the ring is given by

$$\tau \geq \max\{\tau_i, \tau_b, \tau_f\},$$

where τ_b is the lower bound on the cycle time imposed by the movement of holes through the ring, and τ_f is the lower bound imposed by the movement of messages.

Consider a ring of 3 processes, each with $f = 2$ and $b = 5$. Let the internal cycle time of each process be 14. Let there be one message on the ring. Such a ring is shown in Figure 1.5. The lower bounds on the cycle time of the ring are $\tau_i = 14$, $\tau_f = \frac{3*2}{1} = 6$ and $\tau_b = \frac{3*5}{\frac{3}{2}-1} = 30$. Thus the cycle time of the ring is 30.

Consider adding a process to the ring, with $f = 2$, $b = 5$, and internal cycle time 14. The lower bounds on the cycle time of the ring are now $\tau_i = 14$, $\tau_f = \frac{4*2}{1} = 8$ and $\tau_b = \frac{4*5}{\frac{4}{2}-1} = 20$. Thus, the cycle time of the ring is 20.

It is easily seen that adding further processes, until there are 7 processes, on the ring reduces the cycle time to 14. At this point, adding an additional process will increase the cycle time 16. Table 1.1 shows how the cycle time of such a ring changes with the number of processes on the ring.

1.2 Background

Traditionally, VLSI circuits are designed using a single global clock to synchronize various circuit components. Circuits designed without a global clock are said to be *asynchronous*. The class of circuits that do not make any timing assumptions has been shown to be limited [16]. In particular, such circuits are not Turing-complete. There are many classes of asynchronous circuits that make different timing assumptions. Of

Number of Processes	τ_f	τ_b	τ
3	6	30	30
4	8	20	30
5	10	$16\frac{2}{3}$	$16\frac{2}{3}$
6	12	15	15
7	14	14	14
8	16	$13\frac{1}{3}$	16
9	18	$12\frac{6}{7}$	18
10	20	$12\frac{1}{2}$	18

Table 1.1: Change in cycle time of ring with the number of processes on the ring.

these, the class of quasi delay-insensitive(QDI) circuits makes the most conservative timing assumption. These circuits have been shown to be Turing-complete [13]. Such circuits are insensitive to the delays of all gates and most wires (the only timing assumption is the isochronic fork [15] on some wires). QDI circuits are designed as a collection of *processes* that share information via message passing. The processes do not share any variables except those used to implement communication channels. Communication channels are implemented using delay insensitive handshake protocols such as the *four phase handshake* [17].

The four phase handshake protocol is now described. Each communication channel is implemented as a set of wires. The communication channels are unidirectional. A subset of the wires that implement the channel can be assigned to by the sending end of the channel. The receiving end of the channel can only read values on these wires. Similarly, the receiving end of the channel can only assign values to the remaining wires. The sending end can only read the values on these wires. All wires are initialized to a neutral state. The sender begins by changing the values of its output wires to encode a valid data value. The receiver waits for a valid value on its input wires. When it detects a valid value, it changes its output wires to a valid value. The sender waits for this acknowledgment from the receiver, and then resets its outputs to a neutral value. When the receiver observes the neutral value, it resets its outputs to the neutral state. The communication actions on each channel are blocking, that is a process cannot begin the next communication on the same channel until the previous

handshake is complete.

The *cycle time* of a circuit is the mean time between successive occurrences of a particular transition in the circuit. This is a measure of the average rate at which the circuit can process data. Whilst the logical correctness of a QDI circuit is independent of all implementation delays (except at isochronic forks), it is often desirable to minimize the cycle time of a QDI circuit. Oftentimes, the cycle time of a circuit can be much greater than that of the process with the largest cycle time. It has been observed that insertion of additional processes, that act as buffers, on certain channels can greatly reduce a circuit's cycle time [26, 12], as illustrated in Section 1.1

1.3 Problem Definition

In this section, the slack matching problem is defined. The section begins with the definition of a QDI circuit. Next, buffers are defined and finally, slack matching is defined.

A QDI circuit is a pair of a set of processes, $\mathcal{P}$, and a set of channels between processes, $\mathcal{X} \subseteq \mathcal{P} \times \mathcal{P}$. The interface between a process and a channel is called a port. Each process can be described in three ways:

- as a CHP program [17](see appendix A.1 for a brief description of CHP),
- as a handshaking expansion (HSE) [17](see appendix A.2 for a brief description of HSE),
- as a production rule set(PRS) [17](see appendix A.3 for a brief description of PRS).

An *LR-buffer* is a process with the CHP $*[L?; R!]$. A circuit is said to be slack elastic if and only if an arbitrary number of LR-buffers can be inserted on any communication channel in the circuit without affecting its logical correctness [14].

Before defining a slack matching buffer, the concept of a buffer is defined. A buffer is a process, that after a set of initial actions, repeatedly performs communication

actions on a set of ports, called input ports, before performing communication actions on its remaining ports, called output ports. A buffer may manipulate the data received on its input ports before forwarding it on its output ports.

Definition 1.1 (Buffer)**.** *A buffer is a process in a slack elastic circuit that can be written in CHP as*

$$P_{init}; *[P_{loop}]$$

where both P_{init} and P_{loop} are terminating programs and there is at most one communication on each port in any execution of either P_{init} or P_{loop}. Furthermore, for the process to be a buffer, it must be possible to partition its ports into two sets, $\mathcal{I}$ and $\mathcal{O}$, satisfying the following conditions.

- *P_{init} contains only assignments or communications on ports in $\mathcal{O}$.*
- *In any trace of P_{loop} all communications on ports in $\mathcal{I}$ precede all communications on ports in $\mathcal{O}$. $\mathcal{I}$ is said to be the set of* input *ports of the process and $\mathcal{O}$ the set of* output *ports.*
- *An input port of a process can only be connected to an output port of another process.*

Slack matching is an optimization applied to a QDI circuit that reduces the circuit's cycle time below a specified target, via the insertion of LR-buffers on certain channels in the circuit, whilst minimizing a specified cost function. Slack matching can only be performed on circuits that are *slack elastic*. Formally, the slack matching problem can be stated as follows.

Definition 1.2 (Slack Matching Optimization Problem(SMOP))**.** *Let $(\mathcal{P}, \mathcal{X})$ be a slack elastic QDI circuit, $\tau_0 \in \mathbb{R}^+$ be a target cycle time, and S be the LR-buffer that is to be inserted on a subset of the circuit's channels in order to ensure the cycle time of the resulting circuit is at most τ_0. Furthermore, let $C : \mathcal{X} \mapsto [0, \infty)$ be a function that maps each channel to the cost of inserting an additional buffer on this channel. Determine whether there exists a mapping $N : \mathcal{X} \mapsto \mathbb{N}$ such that the circuit obtained*

by inserting $N(x_i)$ instances of S on each channel $x_i \in \mathcal{X}$ has cycle time at most τ_0. If such a mapping exists, determine a mapping such that the total cost of the inserted buffers, $\sum_{x_i \in \mathcal{X}} N(x_i)C(x_i)$, is minimized.

1.4 Another Example

Before continuing further, I explain informally why the need for slack matching arises. In order to do so, the notion of static slack is introduced. The example presented at the beginning of this chapter is analyzed in more detail in this section.

A *pipeline* of LR-buffers is a sequence of LR-buffers such that the L port of the $(i+1)^{th}$ buffer is connected to the R port of the i^{th} buffer. The L port of the first process is said to be the input port of the pipeline. Similarly, the R port of the last process is said to be the output port of a pipeline. The *static slack* of a pipeline of LR-buffers is the maximum number of communications that can be completed on the input port with none being performed on the output port. If a pipeline of n identical LR-buffers has static slack $\frac{n}{2}$, each buffer has static slack $\frac{1}{2}$. A buffer is called a *half buffer* if projecting the buffer onto any pair of input and output ports results in an LR-buffer with static slack $\frac{1}{2}$ (projection is a technique used in the analysis on concurrent systems whereby the behavior of a process is analyzed in terms of its actions on a subset of its variables and channels). If the resulting LR-buffer has static slack 1, the buffer is a *full buffer*.

Let there be a communication channel, implemented via a delay insensitive handshake protocol, in a QDI circuit between processes Q and R. There must be a set of one or more variables, assigned to exclusively by Q, that are used to signal that Q has started a handshake. Similarly, there is a set of one or more variables assigned to exclusively by R that signal that R has started the handshake. Thus in any execution of the circuit where the communication channel is used, Q must signal the start of a handshake. R must wait for this signal from Q before proceeding to acknowledge Q by changing the value of one of its handshake variables.

Recall that a circuit has finite size, however most circuits can have infinitely long

executions. Thus, a finite set of assignments must occur repeatedly in an execution of the circuit. Graphically, the dependences between the occurrences of these assignment can be represented as follows. Each assignment to a variable is represented by a vertex of the graph. Let a directed edge (u, v) between two assignments denote that assignment v cannot occur until the preceding assignment u has occurred. Figure 1.3 shows these dependencies for a pair of processes Q and R that communicate via the four phase handshake protocol on two wires.

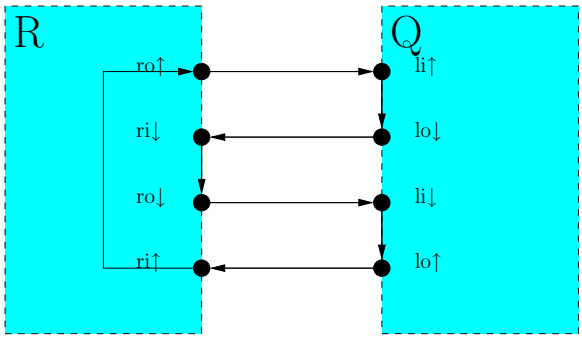


Figure 1.3: Dependences of four phase handshake.

Each process in a circuit has finite size and therefore must have finite static slack. Thus, there must be some dependences between the variables that implement input and output channels of a circuit. Figure 1.4 shows the dependences for a commonly used buffer, the precharged half buffer(PCHB).

For a circuit to be deadlock free, after initialization, at least one assignment must be able to execute without waiting for any other assignments. For an edge (u, v), if the first occurrence of assignment v can execute without waiting for the first occurrence of assignment u, the edge is marked with a solid rectangle. In a physical realization of a circuit, assignments are not instantaneous. Each edge, (u, v), is labeled with a delay, δ, such that assignment v can occur only δ time units after u. Summing the delays along a cycle in the graph and dividing by the number of edges (u, v) on the cycle marked with a solid rectangle provides a lower bound on the circuit's cycle time. This analysis of a circuit's cycle time is formalized in Chapter 2.

Figure 1.5 shows an example of a circuit where the cycle time of all cycles entirely within a process, and the cycle time of cycles between adjacent processes are at most

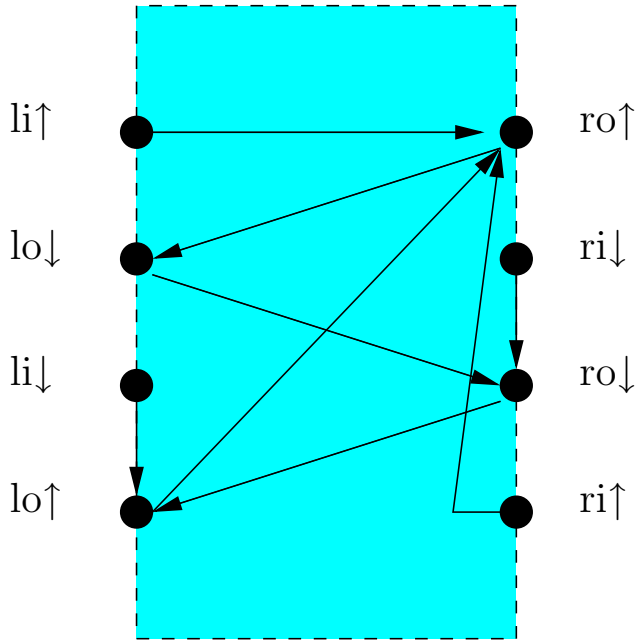


Figure 1.4: Dependences of a PCHB.

14. However, when the processes are connected together as shown, the cycle time of the highlighted cycle is 30. Edges in the figure not marked with a number have delay 0.

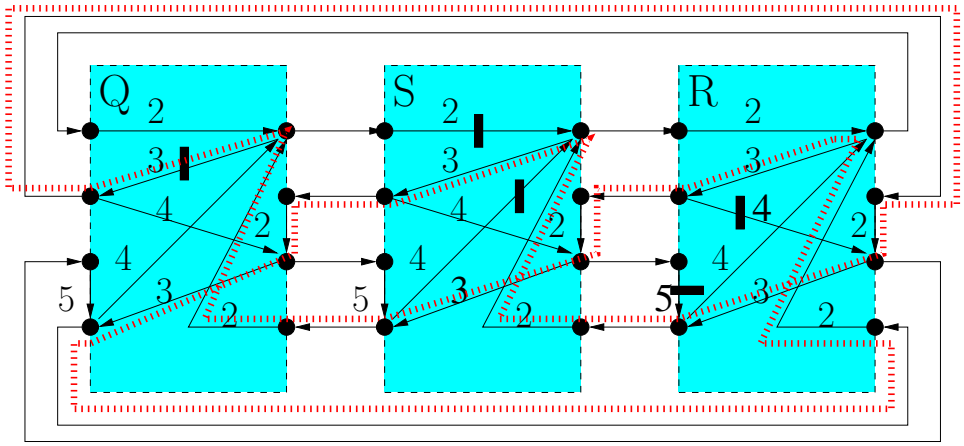


Figure 1.5: Circuit whose cycle time exceeds that of each individual process.

In this example, each of the processes Q, S and R is an LR-buffer. All processes repeatedly alternate between a communication on the input port and a communication on the output port. Process S begins with a communication on its output port. Process Q and R begin with a communication on their input ports. The graph of

processes Q and R differ slightly since R needs to be initialized differently from Q because S begins with a communication on its output port. All communications are implemented using a four phase handshake protocol on two wires. In Figure 1.5, the input port of each process is on its left side and its output ports on the right side.

The processes Q, S and R are connected in a ring. The ring contains one message since only process S can communicate on its output port before a communication on its input port. A message moves around this ring from an input port of a process to an output port of the process. Each process has finite static slack. The number of holes in the ring is the difference between the static slack and the number of messages in the ring. A message can only move into a process if the number of messages it contains is less than its static slack, that is the process is able to accept a message. Holes move from an output port of a process to the input port of a process, in the opposite direction to messages. The cycle time of the ring is determined not only by the movement of messages along the ring, but also the movement of holes along the ring. Note that the circuit in Figure 1.5 is limited by the movement of holes along the ring since the highlighted cycle enters each process on an output port and exits on an input port.

Consider a pair of processes, R and Q, with a channel between them, as shown in Figure 1.6. The assignments $R.ro\uparrow$, $R.ri\uparrow$, $R.ro\downarrow$ and $R.ri\downarrow$ obey a four phase

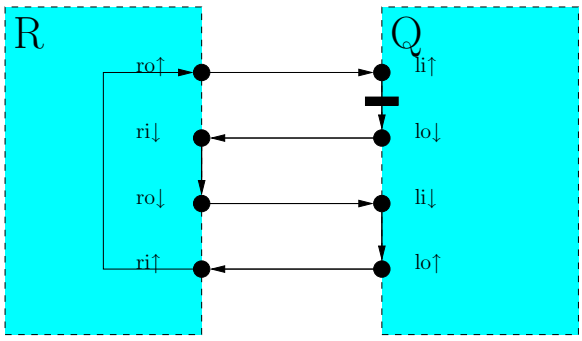


Figure 1.6: Pair of processes with channel between them.

handshake protocol and occur in the sequence

$$R.ro\uparrow; R.ri\downarrow; R.ro\downarrow; R.ri\uparrow; \ldots \quad .$$

Similarly, the assignments $Q.li\uparrow$, $Q.lo\uparrow$, $Q.li\downarrow$ and $Q.lo\downarrow$ obey a four handshake protocol and occur in the sequence

$$Q.li\uparrow; Q.lo\downarrow; Q.li\downarrow; Q.lo\uparrow; \ldots \quad .$$

When there is no buffer between the two processes, the only possible interleaving of these assignments is

$$Q.lo\downarrow; R.ri\downarrow; R.ro\downarrow; Q.li\downarrow; Q.lo\uparrow; R.ri\uparrow; R.ro\uparrow; Q.li\uparrow; \ldots \quad .$$

Inserting a buffer between the processes Q and R, permits other interleavings of the sets of assignments $\{R.ro\uparrow, R.ri\uparrow, R.ro\downarrow, R.ri\downarrow\}$ and $\{Q.li\uparrow, Q.lo\uparrow, Q.li\downarrow, Q.lo\downarrow\}$. An example of such an interleaving is:

$$B.lo\downarrow; R.ri\downarrow; R.ro\downarrow; B.li\downarrow; Q.lo\downarrow; B.ri\downarrow; B.ro\downarrow;$$
$$B.lo\uparrow; R.ri\uparrow; R.ro\uparrow; B.li\uparrow; Q.li\downarrow; Q.lo\uparrow; B.ri\uparrow; B.ro\uparrow; Q.li\uparrow; \ldots \quad .$$

Note that the first $R.ro\downarrow$ occurs before the first $Q.lo\downarrow$. In fact, the i^{th} occurrence of $R.ro\downarrow$ only waits for the $(i-1)^{th}$ occurrence of $Q.lo\downarrow$. This allows, R to proceed without waiting for Q to complete the handshake, the buffer decouples the handshakes of Q and R. Figure 1.7 shows the same circuit with a half buffer inserted between the pro-

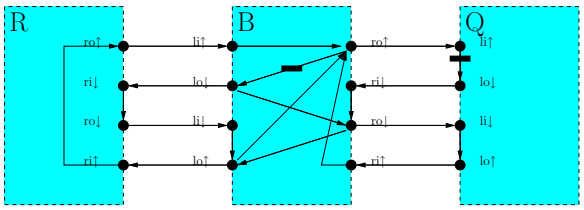


Figure 1.7: Pair of processes with a LR-buffer between them.

cesses R and Q. Inserting the buffer, B, increases the set of possible interleavings of the sets of assignments $\{R.ro\uparrow, R.ri\uparrow, R.ro\downarrow, R.ri\downarrow\}$ and $\{Q.li\uparrow, Q.lo\uparrow, Q.li\downarrow, Q.lo\downarrow\}$. The assignments in this circuit can be interleaved in any manner such that:

- the sets of assignments $\{R.ro\uparrow, R.ri\uparrow, R.ro\downarrow, R.ri\downarrow\}$ and $\{B.li\uparrow, B.lo\uparrow, B.li\downarrow, B.lo\downarrow\}$ have the following interleaving

$$B.lo\downarrow; R.ri\downarrow; R.ro\downarrow; B.li\downarrow; B.lo\uparrow; R.ri\uparrow; R.ro\uparrow; B.li\uparrow; \ldots,$$

and

- the sets of assignments $\{Q.li\uparrow, Q.lo\uparrow, Q.li\downarrow, Q.lo\downarrow\}$ and $\{B.ro\uparrow, B.ri\uparrow, B.ro\downarrow, B.ri\downarrow\}$ have the following interleaving

$$Q.lo\downarrow; B.ri\downarrow; B.ro\downarrow; Q.li\downarrow; Q.lo\uparrow; B.ri\uparrow; B.ro\uparrow; Q.li\uparrow; \ldots,$$

and

- the sets of assignments $\{B.li\uparrow, B.lo\uparrow, B.li\downarrow, B.lo\downarrow\}$ and $\{B.ro\uparrow, B.ri\uparrow, B.ro\downarrow, B.ri\downarrow\}$ are interleaved in the manner specified by the graph of B.

Note that inserting B, replaces a set of edges(that each have delay zero and are not marked with solid rectangles) by a set of new vertices and edges. The new graph has paths between vertices that did not exist prior to the insertion of B. These paths have non-zero delay. Furthermore, some of these paths are marked with solid rectangles. This changes the set of cycles in the circuit's graph. Informally, the buffer allows R and Q to proceed further along in their executions before having to wait on each other. Consider the ring shown in Figure 1.5. Recall that a message cannot move from the input of process R to the output of R unless there is a hole that can move from the output of R to the input of R. The buffers inserted by slack matching are simple LR-buffers that do not start with an initial communication. Thus, inserting a buffer on a ring of processes increases the number of holes on the ring, without changing the number of messages. Inserting a buffer on a ring of processes increases the total delay of a hole moving around the ring. However, this delay is such that the average rate at which holes move around the ring is increased.

Figure 1.8 shows the circuit from Figure 1.5 with a buffer inserted on the channel between processes R and Q. The buffer, B, is exactly like the process Q. Adding the buffer reduces the cycle time to 20. The cycle with the largest cycle time is highlighted in the figure.

Figure 1.9 shows how the cycle time of this circuit changes if additional buffers are introduced on the channel between R and Q. The logical correctness of a slack

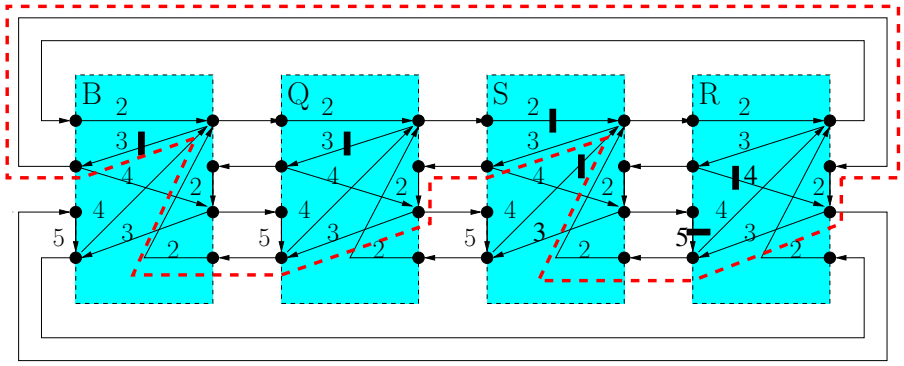


Figure 1.8: Adding a buffer reduces cycle time.

elastic circuit is guaranteed only when additional buffers are introduced. Removing buffers may introduce deadlock [14]. Thus slack matching is restricted to only adding buffers to a circuit. The graph in Figure 1.9 shows that it is not always possible to reduce a system's cycle time via slack matching. When the cycle time of the ring

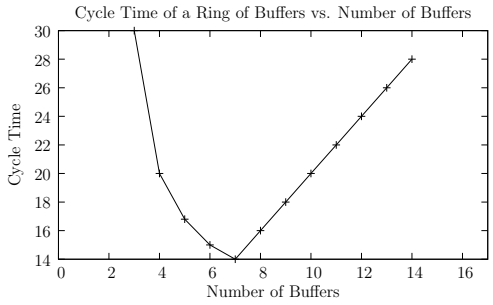


Figure 1.9: Relationship between cycle time of a ring of half buffers and the number of half buffers on the ring.

is limited by the number of holes in the ring, adding buffers(and thus holes) to the ring reduces its cycle time. At some point, adding more buffers to the ring limits the speed at which a message can travel around the ring. In this case, adding additional buffers increases the cycle time of the ring since the additional buffers only slow down the progress of messages around the ring.

1.5 Prior Work

Wong [27] describes a method for slack matching circuits comprised of processes that conform to the precharged half buffer (PCHB) circuit template [12]. When all processes in a circuit can be characterized by identical values of parameters in Wong's model, she formulates in polynomial time a mixed integer linear program(MILP) that is equivalent to the slack matching problem. However, when the parameters of each process in Wong's model differ, she formulates an equivalent MILP in time exponential in the number of processes. Beerel et al. [2] describe a method for slack matching circuits that are comprised of full buffers. However, their model cannot be extended to half buffers. Penzes [22] describes an execution model for QDI circuits described as PRS and for specific circuits formulates a MILP that is equivalent to slack matching. However he does not explain how to do this in general. Slack matching has often been compared to the retiming problem for synchronous circuits. Leiserson et al. [11] describe an efficient algorithm for retiming. Retiming is the analogous problem to slack matching for synchronous circuits. This problem can be solved in polynomial time. A detailed comparison between retiming and slack matching is provided in Section 7.3.

1.6 Contributions

The primary contribution of this thesis is a general method for slack matching slack elastic QDI circuits. It is shown how any slack elastic circuit that can be written as a stable and non-interfering PRS with stable disjuncts can be slack matched using this method provided that all communication channels are implemented using a four phase handshake. The only restrictions placed are on the class of LR-buffers used to slack match the circuit. This method is applicable to circuits that implement communication channel using other handshake protocols as well. The specific examples of the construction of a MILP equivalent to slack matching described by Penzes [22] are special cases of the general method in this thesis. For circuits composed entirely

of half buffers, a MILP that is a generalization of the one described by Wong [27] is derived. This MILP for slack matching is applicable to circuits composed of a larger class of half buffers than described by Wong. In particular, the processes need not be PCHBs. It is shown how this MILP can be generated in polynomial time even when the parameters of each half buffer in the circuit differ. This formulation of slack matching as a MILP, for circuit comprised of half buffers, is analogous to the formulation for full buffers described by Beerel et al. [2].

A second contribution of this thesis is an analysis of the NP-completeness of slack matching. It has been conjectured [9] that slack matching is NP-complete, however there is no proof of this in the literature. It is proven that slack matching is NP-complete via a reduction from subset sum.

1.7 Outline

In Chapter 2, I describe Event-Rule(ER) system [3]. In the remainder of the thesis, I use ER systems to determine the cycle time of a QDI circuit. In Chapter 3, I derive a MILP equivalent to slack matching for a specified class of half buffers. In Chapter 4 I show that slack matching circuits comprised of such half buffers is an NP-complete problem. In Chapter 5, I derive a MILP that is equivalent to slack matching for any slack elastic QDI circuit that does not have unstable disjuncts, provided that the buffer used for slack matching satisfies the specified conditions. In Chapter 6, practical issues in slack matching circuits are discussed. It is also explained how slack matching may be performed by hand. In Chapter 7, the relationship between slack matching and similar problems in circuit design are described. I conclude in Chapter 8 with a summary of the results and directions for future work.

Chapter 2

Event-Rule Systems

Event-Rule(ER) systems can be used to simulate QDI circuits [3, Chapter 2]. In the remainder of this thesis, the cycle time of QDI circuits is determined using ER systems. In this chapter, some definitions and properties of ER systems are stated. The majority of results about ER systems in this chapter are due to Burns [3], and are stated without proof. The only new notions introduced in the chapter are critical delay, critical path and critical cycle.

First, (general) ER systems, whose size may be unbounded, are defined. Since most circuits are designed to operate indefinitely, their ER system has unbounded size. However, since circuits are of finite size, they must repeatedly perform a specified set of actions. Repetitive ER systems are defined in Section 2.2. Repetitive ER systems are unable to simulate circuits that perform an initial set of actions before repeatedly performing a specified set of actions. In Section 2.3, pseudo-repetitive ER systems are defined. Pseudo-repetitive ER systems can simulate the behavior of such circuits. Finally, Section 2.4 describes how the repetitive ER system for a specified class of processes can be generated.

2.1 General ER Systems

Definition 2.1 (Event-Rule System)**.** *An event-rule(ER) system is a pair* $\langle E, R \rangle$ *where:*

- *E is a set of events, and*

- *R is a set of rules defining timing constraints between events. Each rule, $r \in R$ is written $e \xrightarrow{\delta} f$, where*
  - *$e \in E$ is the* source *of r,*
  - *$f \in E$ is the* target *of r, and*
  - *$\delta \in [0, \infty)$ the delay of r.*

Neither E nor R need be finite, however, every event must be the target of a finite number of rules.

Intuitively, each rule $e \xrightarrow{\delta} f$ specifies the timing constraint that event f can only occur δ time units after event e. If there are multiple rules with the same target, f, f can only occur if the timing constraint imposed by all such rules is satisfied. The following definition of timing function formally captures this notion.

Definition 2.2 (Timing Function)**.** *A timing function, t, of an ER system, $\langle E, R\rangle$ is any function $t : E \mapsto [0, \infty)$ such that $t(f) \geq t(e) + \delta$ for each $e \xrightarrow{\delta} f \in R$.*

An ER system can have many timing functions. Since hardware is eager, that is an event will occur as soon as it can occur, the timing function of most interest is one that lets each event occur as early as possible.

Definition 2.3 (Timing Simulation)**.** *A timing simulation, $\hat{t}$, of an ER system,$\langle E, R\rangle$, is the timing function such that for any other timing function t of the ER system, $\hat{t}(e) \leq t(e) \forall e \in E$.*

2.2 Repetitive ER Systems

Many ER systems of unbounded size can be generated from bounded structures. In particular, for many systems there is a finite set of transitions, E', that occurs repeatedly. Repetitive ER systems can be used to analyze such systems.

Definition 2.4 (Repetitive ER System)**.** *A repetitive ER system is a pair $\langle E', R'\rangle$ such that*

- *E' is a finite set of transitions, and*
- *R' is a finite set of repeated rules. Each repeated rule, $r' \in R'$ is written $\langle u, i-o\rangle \xrightarrow{\delta} \langle v, i\rangle$ where:*
  - *u is the source transition of r',*
  - *v is the target transition of r',*
  - *δ is the delay of r'*
  - *o is the occurrence index offset of r', and*
  - *i is the occurrence index of the event v.*

Example 2.1. *Consider the ring oscillator shown in Figure 2.1. Initially let $y =$ **true** and $x = z =$ **false**.*

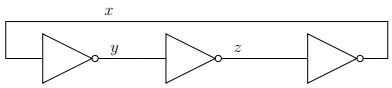


Figure 2.1: Ring Oscillator

Its repetitive ER system is:

$$E' = \{x\uparrow, x\downarrow, y\uparrow, y\downarrow, z\uparrow, z\downarrow\}$$

$$\begin{aligned} R' = \{&\langle z\downarrow, i-1\rangle \xrightarrow{\delta_{x\uparrow}} \langle x\uparrow, i\rangle,\\ &\langle x\uparrow, i-0\rangle \xrightarrow{\delta_{y\downarrow}} \langle y\downarrow, i\rangle,\\ &\langle y\downarrow, i-0\rangle \xrightarrow{\delta_{z\uparrow}} \langle z\uparrow, i\rangle,\\ &\langle z\uparrow, i-0\rangle \xrightarrow{\delta_{x\downarrow}} \langle x\downarrow, i\rangle,\\ &\langle x\downarrow, i-0\rangle \xrightarrow{\delta_{y\uparrow}} \langle y\uparrow, i\rangle,\\ &\langle y\uparrow, i-0\rangle \xrightarrow{\delta_{z\downarrow}} \langle z\downarrow, i\rangle\} \end{aligned}$$

The general ER system, $\langle E, R\rangle$, corresponding to a repetitive ER system $\langle E', R'\rangle$ can be constructed as follows.

$$E = E' \times \mathbb{N}$$

An event $\langle u, i\rangle \in E$ is the indexed occurrence of a transition $u \in E'$. The non-negative integer, i, is called the occurrence index of the event. R is generated by instantiating each repeated rule, $r' \in R'$ for all $i \in \mathbb{N}$ such that $i > o$, where o is the occurrence index offset of r'.

Example 2.2. *Consider the repetitive ER system in Example 2.1. The corresponding general ER system is:*

$$E = E' \times \mathbb{N} = \{\langle x\uparrow, 0\rangle, \langle x\downarrow, 0\rangle, \langle y\uparrow, 0\rangle, \langle y\downarrow, 0\rangle, \langle z\uparrow, 0\rangle, \langle z\downarrow, 0\rangle,$$
$$\langle x\uparrow, 1\rangle, \langle x\downarrow, 1\rangle, \langle y\uparrow, 1\rangle, \langle y\downarrow, 1\rangle, \langle z\uparrow, 1\rangle, \langle z\downarrow, 1\rangle, \dots\}$$

$$\begin{aligned} R = \{&\langle z\downarrow, 0\rangle \xrightarrow{\delta_{x\uparrow}} \langle x\uparrow, 1\rangle,\\ &\langle x\uparrow, 0\rangle \xrightarrow{\delta_{y\downarrow}} \langle y\downarrow, 0\rangle,\\ &\langle y\downarrow, 0\rangle \xrightarrow{\delta_{z\uparrow}} \langle z\uparrow, 0\rangle,\\ &\langle z\uparrow, 0\rangle \xrightarrow{\delta_{x\downarrow}} \langle x\downarrow, 0\rangle,\\ &\langle x\downarrow, 0\rangle \xrightarrow{\delta_{y\uparrow}} \langle y\uparrow, 0\rangle,\\ &\langle y\uparrow, 0\rangle \xrightarrow{\delta_{z\downarrow}} \langle z\downarrow, 0\rangle,\\ &\langle z\downarrow, 1\rangle \xrightarrow{\delta_{x\uparrow}} \langle x\uparrow, 2\rangle,\\ &\langle x\uparrow, 1\rangle \xrightarrow{\delta_{y\downarrow}} \langle y\downarrow, 1\rangle,\\ &\langle y\downarrow, 1\rangle \xrightarrow{\delta_{z\uparrow}} \langle z\uparrow, 1\rangle,\\ &\langle z\uparrow, 1\rangle \xrightarrow{\delta_{x\downarrow}} \langle x\downarrow, 1\rangle,\\ &\langle x\downarrow, 1\rangle \xrightarrow{\delta_{y\uparrow}} \langle y\uparrow, 1\rangle,\\ &\langle y\uparrow, 1\rangle \xrightarrow{\delta_{z\downarrow}} \langle z\downarrow, 1\rangle,\\ &\dots\} \end{aligned}$$

A timing function of a repetitive ER system is a function that is a timing function of the corresponding general ER system. A linear timing function of a repetitive ER system is one that satisfies the following property.

Definition 2.5 (Linear Timing Function)**.** *A linear timing function,* $\bar{t} : E' \times \mathbb{N} \mapsto [0, \infty)$*,of a repetitive ER system* $\langle E', R' \rangle$*, is a timing function such that for each* $v \in E'$*, there exist* x_v *and* p_v *such that*

$$\bar{t}(v, i) = x_v + p_v \cdot i \quad \forall v \in E', i \in \mathbb{N}$$

For a specific linear timing function, $\bar{t}$, p_v is called the *cycle period* of transition v in $\bar{t}$ and x_v the offset of v in $\bar{t}$. Note that each x_v and p_v is independent of i.

A repetitive ER system, $\langle E', R' \rangle$, can be represented as a directed graph $G = (N, A)$. N is the set of nodes, and is equal to E'. A is the set of arcs such that $(u, v) \in A$ if and only if $\langle u, i - o \rangle \xrightarrow{\delta} \langle v, i \rangle \in R'$. Let $\Delta : A \mapsto [0, \infty)$ map each arc to the delay of the corresponding rule and $O : A \mapsto \mathbb{N}$ map each arc to the occurrence index offset of the corresponding rule. This graph is called the *collapsed constraint graph* of $\langle E', R' \rangle$.

In a given timing function, all transitions in the same strongly connected component of a collapsed constraint graph have the same cycle period. The collapsed constraint graph of typical asynchronous circuits is strongly connected [3, 10]. Following Burns [3], only repetitive ER systems whose collapsed constraint graph is strongly connected are considered. *Thus, all transitions have the same cycle period* p *in a given timing function.* This cycle period is the cycle period of the timing function.

Burns has shown that there exists a linear timing function of a repetitive ER system with minimal cycle period.

Definition 2.6 (Minimum Period Linear Timing Function(MPLTF))**.** *A minimum period linear timing function(MPLTF) of a repetitive ER system is the linear timing function of the repetitive ER system with minimal period.*

Burns has shown that the period of the MPLTF of a repetitive ER system is an approximation of the timing simulation of the corresponding timing simulation. In particular, he has shown that the period of a MPLTF is arbitrarily close to the mean time between successive occurrences of a transition in the timing simulation.

Definition 2.7 (Cycle Time)**.** *The cycle time of a QDI circuit is the minimum cycle period of the repetitive ER system that simulates the circuit.*

2.2.1 Verifying the Cycle Period of a Repetitive ER System

In this section, I state how it can be verified that a repetitive ER system has a specific cycle time, $p \in \mathbb{R}^+$. In order to do so, the notion of critical delay of a repeated rule is introduced. The critical delay as defined depends on the value of p. Thus, the phrase *critical delay at* p specifies the value of p used in the computation of the critical delay.

The minimum cycle period of a repetitive ER system, $\langle E', R' \rangle$ is at most p, if there exists a linear timing function with period p. That is, for each transition $e \in E$ there exists x_e such that the following inequality is satisfied for each rule $\langle u, i - o \rangle \stackrel{\delta_{uv}}{\rightarrow} \langle v, i \rangle \in R'$.

$$x_v - x_u \geq \delta_{uv} - o_{uv} \cdot p$$

Each x_e corresponds to the offset in a linear timing function. Rewriting in matrix form, the minimum cycle period of $\langle E', R' \rangle$ is at most p if there exists $\mathbf{x}$ such that

$$\mathcal{A}\mathbf{x} \geq \mathbf{y} \tag{2.1}$$

$$\mathbf{x} \geq 0 \tag{2.2}$$

where

- $\mathbf{y} \in \mathbb{R}^{|R'|}$ such that for each rule $\langle u, i - o_j \rangle \stackrel{\delta_j}{\rightarrow} \langle v, i \rangle \in R'$, $y_j = \delta_j - p \cdot o_j$,
- 0 is the zero vector of length $|E'|$,
- $\mathbf{x} \in \mathbb{R}^{|E'|}$ is such that each entry x_u corresponds to the offset of u in a linear timing function,

- $\mathcal{A}$ is a $|R'| \times |E'|$ matrix such that for each rule, $r_j \in R'$

$$\mathcal{A}_{j,k} = \begin{cases} -1 & \text{if } e_k \text{ is the source transition of } r_j, \\ 1 & \text{if } e_k \text{ is the target transition of } r_j, \\ 0 & \text{otherwise} \end{cases}$$

Since this set of inequalities is a system of difference constraints, if $\mathbf{x}$ satisfies inequality (2.1), then so does any $\mathbf{x}' : \mathbf{x}' = \mathbf{x} + d1, d \in \mathbb{R}$ where 1 is a unit vector. It can be shown that the system of difference constraints (2.1) can be satisfied if and only if there are no positive weight cycles in a directed graph that has arc-node incidence matrix $\mathcal{A}$ and arc weights $\mathbf{y}$ [4]. By definition, $\mathcal{A}$ is the arc-node incidence of the collapsed constraint graph of $\langle E', R' \rangle$.

A directed path in a directed graph is a sequence of arcs $\{(u_i, v_i)\}$ such that $u_i = v_{i-1}$ for all $i > 1$. Let $\|\pi\|$ denote the number of arcs in directed path π. A cycle is a path, c, such that $v_{\|c\|} = u_1$. Thus, a circuit has cycle time at most p if and only if for each cycle C in its collapsed constraint graph,

$$\sum_{(u,v) \in C} \delta_{uv} - p \cdot o_{uv}$$

where δ_{uv} is the delay of rule corresponding to arc to (u, v) and o_{uv} is the occurrence index offset of this rule. This value is defined as the critical delay of the cycle at p. The critical delay of an arc at p is the contribution of the arc to the critical delay of any cycle that traverses the arc.

Definition 2.8 (Critical Delay)**.** *Let $r' \in R'$ be the repeated rule $\langle u, i - o_{uv} \rangle \xrightarrow{\delta_{uv}} \langle v, i \rangle$. For $p \in [0, \infty)$, the critical delay of r' at p is $\delta_{uv} - p \cdot o_{uv}$.*

The critical delay at p of an arc in a collapsed constraint graph is the critical delay at p of the corresponding repeated rule. The critical delay at period p of a directed path, π, in a collapsed constraint graph is the sum of the critical delays at p of the arcs on the path.

A critical path at p between two nodes of a collapsed constraint graph is a path with maximum critical delay at p between the nodes. A critical cycle at p is a cycle in the collapsed constraint graph with maximum critical delay at p.

Thus, the minimum period of the repetitive ER system corresponding to a collapsed constraint graph is at most p if and only if all critical cycles at p have non-positive critical delay

Example 2.3. *Figure 2.2 show a fragment of a collapsed constraint graph. Each edge in the graph is marked with its' delay. An edge marked with a solid rectangle has occurrence index offset 1. All other edges have occurrence index offset 0. There are two paths between vertices u and x. At cycle time 14, the path* $(u, v)(v, x)$ *has critical delay* $14 - 14 \cdot 1 = 0$. *The path* $(u, w)(w, x)$ *has critical delay* $6 - 14 \cdot 0 = 6$. *Thus,* $(u, w)(w, x)$ *is the critical path between u and x.*

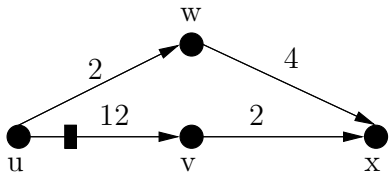


Figure 2.2: Critical paths.

2.3 Pseudo-Repetitive ER Systems

Whilst many circuits can be modeled as repetitive ER systems, many circuits of interest consist of a finite set of initial events followed by a set of repeated transitions. Such circuits 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 [3] shows that the cycle time of a pseudo-repetitive ER system is approximated by that of the repetitive system (E_1', R_1').

2.4 Constructing a Repetitive ER System

In this Section, I state how to generate a repetitive ER system from the description of a QDI circuit. These constructions are from Burns [3][Chapter 4]. In Section 2.4.1 I state how to construct the repetitive ER system for a circuit such that all processes are described as HSE. In Section 2.4.2 I state how to construct the repetitive ER system for a circuit such that all processes are described as PRS.

2.4.1 HSE

I begin with the construction of repetitive ER systems from straight line HSE in standard form. This construction is then generalized to straight line HSE.

A *straight line handshaking expansion(SLHE)* is a HSE such that each selection statement is of the form $[C \rightarrow \textbf{skip}]$ with C being a conjunction of literals, and each repetition statement is of the form $[\textbf{true} \rightarrow S]$. A SLHE has a vacuous wait on a literal if the literal l is **true** in the initial state and as the process executes, a wait $[l \wedge C]$ is encountered without l having become **false**. A SLHE has a vacuous assignment if there is an assignment that does not change the state of the program when it is encountered during execution. A SLHE has a repeated assignment if the same repetition statement contains two assignments of the same value to the same variable.

A collection of SLHE in *standard form* is one that satisfies the following conditions.

1. The collection is deadlock free.

2. Each process is of the form $S; *[T]$ and contains no vacuous waits or assignments.

3. Each process has no repeated assignments.

4. S and T are a sequence of alternating waits and assignments.

5. S begins with a wait and ends with a wait.

6. T begins with an assignment and ends with a wait.

7. Each variable appears in one process only and is either a local variable, or a variable implementing a communication action.

8. If a variable, v appears in a wait of one process, v is either assigned to by the process, or is not assigned to in any process and is the input variable of a port with the corresponding output variable only assigned to in this process.

Given a SLHE in standard form, $S; *[T]$, a repetitive ER system $\langle E', R' \rangle$ can be constructed by adding a transition to E' for each assignment that appears in T. In addition, transitions are added for each assignment to each input variable of the process. Let v be an assignment in T. Let u be an assignment that causes a literal in a wait immediately preceding v to evaluate to true. For each such u, add to R' a repeated rule, $\langle u, i - o \rangle \xrightarrow{\delta} \langle v, i \rangle$. If v is the first assignment in T, let the last wait in T be the wait immediately preceding v in T. In either case, u is said to be a *constraining wait* of v. Let u be an assignment such that there is exactly one wait in sequence between u and v in T. For each such u, add a repeated rule $\langle u, i-o \rangle \xrightarrow{\delta} \langle v, i \rangle$ to R'. If v is the first assignment in T, let u be the last assignment in T. In either case, u is said to be a *constraining assignment* of v. The value of the occurrence index offset, o, is the number of times that v must occur in any execution of the HSE before the first occurrence of u.

Example 2.4. *Consider the HSE of the ring oscillator in Example 2.1:*

$$*[x\uparrow; y\downarrow; z\uparrow; x\downarrow; y\uparrow; z\downarrow]$$

Its repetitive ER system is:

$$E' = \{x\uparrow, x\downarrow, y\uparrow, y\downarrow, z\uparrow, z\downarrow\}$$

$$\begin{aligned} R' = \{&\langle z\downarrow, i-1\rangle \overset{\delta_{x\uparrow}}{\rightarrow} \langle x\uparrow, i\rangle, \\ &\langle x\uparrow, i-0\rangle \overset{\delta_{y\downarrow}}{\rightarrow} \langle y\downarrow, i\rangle, \\ &\langle y\downarrow, i-0\rangle \overset{\delta_{z\uparrow}}{\rightarrow} \langle z\uparrow, i\rangle, \\ &\langle z\uparrow, i-0\rangle \overset{\delta_{x\downarrow}}{\rightarrow} \langle x\downarrow, i\rangle, \\ &\langle x\downarrow, i-0\rangle \overset{\delta_{y\uparrow}}{\rightarrow} \langle y\uparrow, i\rangle, \\ &\langle y\uparrow, i-0\rangle \overset{\delta_{z\downarrow}}{\rightarrow} \langle z\downarrow, i\rangle\} \end{aligned}$$

The cycle time of this ER system is the minimum p such that the following system of linear inequalities can be satisfied.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_{x\uparrow} \\ x_{x\downarrow} \\ y_{y\uparrow} \\ y_{u\downarrow} \\ z_{z\uparrow} \\ z_{z\downarrow} \end{pmatrix} \geq \begin{pmatrix} \delta_{x\uparrow} - p \\ \delta_{x\downarrow} \\ \delta_{y\uparrow} \\ \delta_{y\downarrow} \\ \delta_{z\uparrow} \\ \delta_{z\downarrow} \end{pmatrix}$$

$$\begin{pmatrix} x_{x\uparrow} \\ x_{x\downarrow} \\ y_{y\uparrow} \\ y_{u\downarrow} \\ z_{z\uparrow} \\ z_{z\downarrow} \end{pmatrix} \geq 0$$

Let P and Q be processes with a communication channel between them. Let u be an output variable of process P that implements a communication action on this

channel. Let v be the corresponding input variable of process Q. Following Burns [3, Chapter 4], for each such pair of variables u and v, a repeated rule with delay and occurrence index offset zero is added between each transition on u and the same transition on v. If the wire connecting the variables has non-zero delay, a fraction of its delay is added to that of all rules whose target is a transition on u. The remaining delay added to that of all rules whose source is a transition on v. That is, the delay of a wire between processes is apportioned in some manner between the two processes.

A collection of SLHE in standard form does not permit any concurrent assignments or concurrent waits. The requirement that S and T in an SLHE be sequences can be relaxed in the following manner.

Consider replacing any assignment by the concurrent composition of one or more assignments. Let the program fragment $\ldots A'; w; A$ be such that A and A' are concurrent assignments to one or more variables. Any assignment that makes a literal in w true is a constraining wait of each assignment in A. Similarly, each assignment in A' is a constraining assignment of each assignment in A.

Consider two general statements composed in parallel. For each program fragment

$$\ldots \ ; ((\ldots ; a'; w'), (\ldots ; a''; w'')); a; \ldots$$

the constraining assignments of a are a' and a''. The constraining waits of a are any assignments that make a literal in either w' or w'' true.

Example 2.5. *Figure 2.3 shows the collapsed constraint graph of the repetitive part of the pseudo-repetitive ER system that describes the HSE:*

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

All edges have an occurrence index offset of zero unless they are marked with a rectangular box which denotes an occurrence index offset of 1. The pseudo-repetitive ER system is:

$$E_0' = \{lo{\downarrow}, ri{\downarrow}, ro{\downarrow}, li{\downarrow}, lo{\uparrow}\}$$

$$E_0 = \{\langle lo{\downarrow}, 0\rangle, \langle ri{\downarrow}, 0\rangle, \langle ro{\downarrow}, 0\rangle, \langle li{\downarrow}, 0\rangle, \langle lo{\uparrow}, 0\rangle\}$$

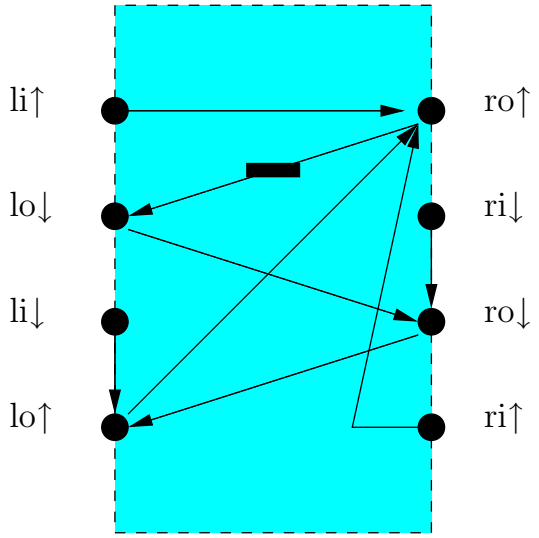


Figure 2.3: Collapsed constraint graph of PCHB.

$$E_1' = \{lo\uparrow, lo\downarrow, li\uparrow, li\downarrow, ri\uparrow, ri\downarrow, ro\uparrow, ro\downarrow\}$$

$$E_1 = \{\langle lo\uparrow, i\rangle | i \geq 1\} \cup \{\langle lo\downarrow, i\rangle | i \geq 1\} \cup \{\langle li\uparrow, i\rangle | i \geq 0\} \cup \{\langle li\downarrow, i\rangle | i \geq 1\} \cup$$
$$\{\langle ro\uparrow, i\rangle | i \geq 0\} \cup \{\langle ro\downarrow, i\rangle | i \geq 1\} \cup \{\langle ri\uparrow, i\rangle | i \geq 0\} \cup \{\langle ri\downarrow, i\rangle | i \geq 1\}$$

$$R_0 = \{\langle lo\downarrow, 0\rangle \xrightarrow{\delta_1} \langle ro\downarrow, 0\rangle$$
$$\langle ri\downarrow, 0\rangle \xrightarrow{\delta_2} \langle ro\downarrow, 0\rangle$$
$$\langle li\downarrow, 0\rangle \xrightarrow{\delta_3} \langle lo\uparrow, 0\rangle$$
$$\langle ro\downarrow, 0\rangle \xrightarrow{\delta_4} \langle lo\uparrow, 0\rangle$$
$$\langle lo\uparrow, 0\rangle \xrightarrow{\delta_5} \langle ro\uparrow, 0\rangle$$
$$\}$$

$$R_1' = \{\langle lo\downarrow, i-0\rangle \xrightarrow{\delta_1} \langle ro\downarrow, i\rangle$$
$$\langle ri\downarrow, i-0\rangle \xrightarrow{\delta_2} \langle ro\downarrow, i\rangle$$
$$\langle li\downarrow, i-0\rangle \xrightarrow{\delta_3} \langle lo\uparrow, i\rangle$$
$$\langle ro\downarrow, i-0\rangle \xrightarrow{\delta_4} \langle lo\uparrow, i\rangle$$
$$\langle lo\uparrow, i-0\rangle \xrightarrow{\delta_5} \langle ro\uparrow, i\rangle$$
$$\langle li\uparrow, i-0\rangle \xrightarrow{\delta_6} \langle ro\uparrow, i\rangle$$
$$\langle ri\uparrow, i-0\rangle \xrightarrow{\delta_7} \langle ro\uparrow, i\rangle$$
$$\langle ro\uparrow, i-1\rangle \xrightarrow{\delta_8} \langle lo\downarrow, i\rangle$$
$$\}$$

The corresponding general ER system is:

$$E = E_O \cup E_1$$

$$
\begin{aligned}
R = & R_0 \cup \\
& \{\langle lo\downarrow, i-0\rangle \xrightarrow{\delta_1} \langle ro\downarrow, i\rangle | i \geq 1\} \cup \\
& \{\langle ri\downarrow, i-0\rangle \xrightarrow{\delta_2} \langle ro\downarrow, i\rangle | i \geq 1\} \cup \\
& \{\langle li\downarrow, i-0\rangle \xrightarrow{\delta_3} \langle lo\uparrow, i\rangle | i \geq 1\} \cup \\
& \{\langle ro\downarrow, i-0\rangle \xrightarrow{\delta_4} \langle lo\uparrow, i\rangle | i \geq 1\} \cup \\
& \{\langle lo\uparrow, i-0\rangle \xrightarrow{\delta_5} \langle ro\uparrow, i\rangle | i \geq 1\} \cup \\
& \{\langle li\uparrow, i-0\rangle \xrightarrow{\delta_6} \langle ro\uparrow, i\rangle | i \geq 0\} \cup \\
& \{\langle ri\uparrow, i-0\rangle \xrightarrow{\delta_7} \langle ro\uparrow, i\rangle | i \geq 0\} \cup \\
& \{\langle ro\uparrow, i-1\rangle \xrightarrow{\delta_8} \langle lo\downarrow, i\rangle | i \geq 1\}
\end{aligned}
$$

2.4.2 PRS

Each process can also be described as a production rule set(PRS). Consider a production rule set derived from a handshaking expansion as described above. Let $G \rightarrow v$ be a production rule in a such a closed PRS, where v is a simple assignment and G is a conjunctive guard. For each assignment v, let u be an assignment that causes a literal in G to evaluate to **true**. For each such u, add to R' a repeated rule $\langle u, i-o\rangle \xrightarrow{\delta} \langle v, i\rangle$. The value of o is the number of times that u must occur before the first occurrence of v. This can be determined either from the HSE, or via simulation.

2.4.3 Disjunctive Circuits

Many circuits cannot be written as collections of HSE or PRS satisfying the restrictions in Section 2.4.1 and 2.4.2. In this section, I state how ER systems may be used to model the behavior of such circuits.

2.4.3.1 Disjunctive HSE

Some collections of HSE that do not satisfy the restrictions of Section 2.4.1 can still be modeled as a collection of HSE satisfying these restrictions. In this Section, such a class of HSE is described.

For a HSE with deterministic selection statements, only one of the guards in the selection statement may be true at any time. When closing the HSE, the environment must be chosen in such a manner that the same guard of each selection statement may evaluate to **true** in any execution. All other guards in the selection statement may be removed since in the particular scenario being considered, they never evaluate to **true**. If the guard is conjunctive, this part of the HSE satisfies the restrictions in Section 2.4.1. If the guard is disjunctive, it can be transformed to conjunctive guard in the manner described below.

Consider a HSE with a wait on a disjunction. If only one of the disjuncts may evaluate to **true** in any execution, and this disjunction is the same for all executions, the other disjuncts may be removed from the wait statement.

If more than one disjunct may evaluate to **true**, the HSE cannot be modeled directly as a repetitive ER system. Such HSE can either be excluded, or modeled in the following way. The disjunction is replaced by a conjunction of the disjuncts that may evaluate to **true** in some execution of the HSE. As with selection statements, if the environment can be chosen to ensure that only one of the disjunct evaluates to **true**, the circuit with such an environment can be modeled as a repetitive ER system. Circuits where this is not possible occur rarely in practice [3, Chapter 4]. These circuits are excluded from consideration.

2.4.3.2 Disjunctive PRS

A circuit described as a PRS is disjunctive if at least one PR has a disjunction in its guard.

There are two types of disjunctions.

- Stable disjunctions are ones such that if any disjunct evaluates to **true**, it remains **true** until the target assignment of the PRS fires.
- Unstable disjunctions are all other disjunctions.

In the remainder of this thesis, only PRS with stable disjuncts are considered. Circuits with data have stable disjuncts. In practice, circuits with unstable disjuncts are

rare [10, 3].

Stable disjuncts can be divided into three cases.

- In the first case, during all executions of the PRS only one disjunct evaluates to **true** when the rule fires effectively. This disjunct is the same across all executions of the PRS. In this case, the remaining disjuncts can be ignored since they can never cause an effective firing.

- In the second case, the disjuncts are mutually exclusive. In this case, when the PRS is closed, the environment of the PRS is chosen in such a manner that only one of the mutually exclusive disjuncts evaluates to **true** in any execution of the PRS. Such disjuncts typically arise due to data dependencies.

- Typically, the preceding cases cover most disjunctions in a circuit. Circuits with neither of the preceding type of stable disjunctions can either be excluded, or the disjuncts in each guard that may evaluate to **true** can be determined via simulation and the disjunction can be replaced by a conjunction of the disjuncts that may evaluate to **true**. In the latter case, the cycle time of the resulting PRS is an upper bound on the cycle time of the original PRS. The disjunction is modeled as a conjunction in order to allow the PRS to be modeled as an ER system.

Given a PRS with an environment that satisfies the preceding restrictions, Lee [10, Chapter 7] shows how the corresponding ER system can be generated.

Chapter 3

Slack Matching Circuits of Half Buffers

In this Chapter, I derive necessary and sufficient conditions to guarantee that a circuit composed of a specified class of half buffers has a cycle time τ_0. These conditions are used to state the slack matching problem as a mixed integer linear program(MILP).

This Chapter is organized as follows.

- First, in Section 3.1.1, the paths in the collapsed constraint graph of half buffers are classified. Based on this classification of paths, the class of half buffers considered in the remainder of the Chapter is specified in Section 3.1.2. In Section 3.1.3, the critical cycles at the target cycle time are determined.

- In Section 3.2.1 process graphs are defined. In Section 3.2.2, undirected cycles in a process graph are related to directed cycles in the collapsed constraint graph of the same circuit. Since the number of undirected cycles in a graph may be exponential in the size of the graph, in Section 3.2.3, a set of sufficient conditions for the circuit described by a process graph to have cycle time less than or equal to a specified target is described. This set of conditions is polynomial in the size of the graph. In Section 3.2.4, it is shown when this set of conditions is necessary for a circuit to have cycle time less than or equal to the a specified target.

- Next, in Section 3.3.1 some properties of a pipeline of LR-buffers, used for slack matching, are stated. Using these properties, in Section 3.3.2, the conditions

derived in Section 3.2.3 are modified in order to introduce a variable number of slack matching buffers on each channel. This results in a mixed integer linear program that can be solved to determine the number of slack matching buffers to add to each channel in a circuit in order for the circuit to have cycle time less than or equal to a specified target. In Section 3.3.3, the time complexity of generating the MILP is determined.

- In Section 3.4, the results of slack matching two different circuits by solving the aforementioned MILP are discussed. This chapter concludes in Section 3.5 with a summary of the results.

3.1 ER Systems of Half Buffers

Recall that the goal of slack matching is to introduce LR-buffers into a circuit is such a manner that the resulting circuit has cycle time at most τ_0. In this Section, the critical delay at cycle time τ_0 of all cycles in the collapsed constraint graph of a circuit comprised of a specified class of buffers is derived.

This Section is organized as follows.

- In Section 3.1.1 the paths in a buffer's collapsed constraint graph are classified.
- Next, in Section 3.1.2 the class of buffers in the circuits considered is specified.
- Finally, the critical delay at τ_0 of all cycles in the collapsed constraint graph of such circuits is determined.

3.1.1 Classification of Paths in a Buffer's Collapsed Constraint Graph

In order to state the assumptions made about the class of buffers considered, the paths in the collapsed constraint graph between transitions of the variables that implement ports need to be classified.

The *input variables* of a process are variables that implement a port such that these variables are not assigned to by the process. The *output variables* of a process are variables that implement a port such that these variables are assigned to by the process. All ports are assumed to implement a four phase handshake. As described in Section 2.4.3 each process can be modeled as having one input variable and one output variable per port. Note that in order to do so for circuits with data, the environment of the circuit must be such that for each communication channel, the same data value is repeatedly sent on the channel. A transition of an input(output) variable is said to be an *input(output) transition*. The transition on the input(output) variable of an input port and the corresponding transitions on the output(input) variable of the output port of the same channel are connected by a rule with delay and occurrence index offset equal to zero. A delay of the wire connecting the two variables can be apportioned as follows. A fraction of this delay can be added to all rules whose target is the output transition of a port. The remaining delay can be add to all rules whose source is the corresponding input transition of the corresponding port. Thus, regardless of the value of τ_0, the critical delay at τ_0 of this rule is zero. Therefore, in the remainder of this section such edges are ignored and the vertices corresponding to such transitions considered to be the same.

In a QDI circuit's collapsed constraint graph, all directed paths between transitions on variables that implement ports are classified by the type of ports that the variables belong to. Let a path, π, begin at an input transition of port y in process u. Let π end at an output transition of port z in process v.

- If y is an input port and z is an output port π is said to be of type Φ. Additionally, if $u = v$ and π traverses only paths in process u, it is said to be of type ϕ. Thus, a ϕ path is a Φ path that traverses exactly one process.

- If y is an output port and z is an input port π is said to be of type B. Additionally, if $u = v$ and π traverses only paths in process u, it is said to be of type β. Thus, a β path is a B path that traverses exactly one process.

- If y is an input port and z is an input port π is said to be of type Λ. Additionally,

if $u = v$ and π traverses only paths in process u, it is said to be of type λ. Thus, a λ path is a Λ path that traverses exactly one process.

- If y is an output port and z is an output port π is said to be of type P. Additionally, if $u = v$ and π traverses only paths in process u, it is said to be of type ρ. Thus, a ρ path is a P path that traverses exactly one process.

Figure 3.1 shows all ϕ, λ, β and ρ paths in a buffer that uses a four phase handshake.

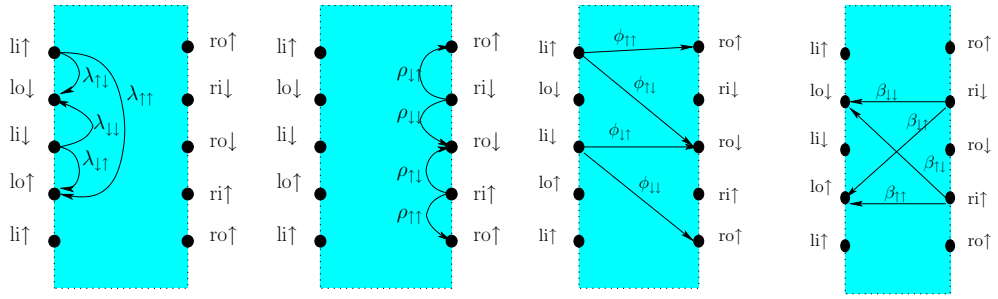


Figure 3.1: Paths in a buffer's collapsed constraint graph.

The notation $t^{ij}_{\uparrow\downarrow}$ is used to denote a path of type $t \in \{\phi, \beta, \rho, \lambda, \Phi, B, \Lambda, P\}$ from transition $w\uparrow$ to transition $x\downarrow$ where w is the input variable of port i and x is the output variable of port j. The notation $t^{ij}_{\uparrow\downarrow}(p)$ is used to denote a path of type $t \in \{\phi, \beta, \rho, \lambda\}$ in the collapsed constraint graph of process p, between an input transition of port i and an output transition of port j. Let $t^{i,j}$ denote the set of all paths of type t between a transition of an input variable of port i and a transition of an output variable of port j.

3.1.2 Assumptions About Processes in a Circuit

The following assumptions are satisfied by all processes in the circuits considered in this chapter.

Assumption 3.1. *Each process is a half buffer.*

Assumption 3.2. *A process communicates on each channel exactly once per iteration.*

For circuits in which a process conditionally communicates on a channel based on data, assumption 3.2 can be satisfied by choosing an environment that ensures that each process exercises the same set of channels on each iteration. This is done as described in Section 2.4.3.

Assumption 3.3. *A port can be initialized in one of three ways.*

1. *An output port may complete a communication action before any input port of the process starts a communication. Such a port is said to have an initial send.*

2. *Consider an output port with an initial send. The port at the other end of a channel containing this port is an input port that is said to have an initial receive.*

3. *A port that has neither an initial send nor an initial receive is said to be neutral.*

Assumption 3.4. *The critical delay, at τ_0 of a path between input and output transitions of process p is less than or equal to the value in Table 3.1, where $f_p^{i,j}$, $b_p^{i,j}$, $l_p^{i,j}$ and $r_p^{i,j}$ are constants.*

In Table 3.1, $i \in N$ denotes that port i is neutral, $i \in S$ that i has an initial send and $i \in R$ that there is an initial receive of i. Since the processes in the circuit are assumed to be half buffers, a process cannot have both a port that has an initial send and a port that has an initial receive.

Assumption 3.5. *The critical delay at τ_0 of any cycle that contains only transitions from one process is less than* 0.

Assumption 3.5 states that target cycle time be at least as large as the largest ratio of the delay along a cycle entirely within a process to the sum of the occurrence index offsets along this cycle. Such cycles are not affected by slack matching since slack matching only adds buffers to channels, it does not change a process in the circuit.

Path	$i,j \in N$	$i \in N, j \in S$	$i \in R, j \in N$	
$\phi^{i,j}_{\uparrow\uparrow}$	$f^{i,j}_p$	$f^{i,j}_p - \tau_0$	$f^{i,j}_p$	
$\phi^{i,j}_{\uparrow\downarrow}$	$f^{i,j}_p - \frac{\tau_0}{2}$	$f^{i,j}_p - \frac{\tau_0}{2}$	$f^{i,j}_p - \frac{\tau_0}{2}$	
$\phi^{i,j}_{\downarrow\uparrow}$	$f^{i,j}_p + \frac{\tau_0}{2}$	$f^{i,j}_p - \frac{\tau_0}{2}$	$f^{i,j}_p - \frac{\tau_0}{2}$	
$\phi^{i,j}_{\downarrow\downarrow}$	$f^{i,j}_p$	$f^{i,j}_p$	$f^{i,j}_p - \tau_0$	
Path	$i,j \in N$	$i \in R, j \in N$	$i \in N, j \in R$	$i,j \in R$
$\lambda^{i,j}_{\uparrow\uparrow}$	$l^{i,j}_p - \frac{\tau_0}{2}$	$l^{i,j}_p - \frac{\tau_0}{2}$	$l^{i,j}_p - \frac{\tau_0}{2}$	$l^{i,j}_p - \frac{\tau_0}{2}$
$\lambda^{i,j}_{\uparrow\downarrow}$	$l^{i,j}_p - \tau_0$	$l^{i,j}_p - \tau_0$	$l^{i,j}_p$	$l^{i,j}_p$
$\lambda^{i,j}_{\downarrow\uparrow}$	$l^{i,j}_p$	$l^{i,j}_p - \tau_0$	$l^{i,j}_p$	$l^{i,j}_p - \tau_0$
$\lambda^{i,j}_{\downarrow\downarrow}$	$l^{i,j}_p - \frac{\tau_0}{2}$	$l^{i,j}_p - \frac{\tau_0}{2}$	$l^{i,j}_p + \frac{\tau_0}{2}$	$l^{i,j}_p - \frac{\tau_0}{2}$
Path	$i,j \in N$	$i \in S, j \in N$	$i \in N, j \in R$	
$\beta^{i,j}_{\uparrow\uparrow}$	$b^{i,j}_p - \frac{\tau_0}{2}$	$b^{i,j}_p + \frac{\tau_0}{2}$	$b^{i,j}_p - \frac{\tau_0}{2}$	
$\beta^{i,j}_{\uparrow\downarrow}$	$b^{i,j}_p - \tau_0$	$b^{i,j}_p$	$b^{i,j}_p$	
$\beta^{i,j}_{\downarrow\uparrow}$	$b^{i,j}_p$	$b^{i,j}_p$	$b^{i,j}_p$	
$\beta^{i,j}_{\downarrow\downarrow}$	$b^{i,j}_p - \frac{\tau_0}{2}$	$b^{i,j}_p - \frac{\tau_0}{2}$	$b^{i,j}_p + \frac{\tau_0}{2}$	
Path	$i,j \in N$	$i \in S, j \in N$	$i \in N, j \in S$	$i,j \in S$
$\rho^{i,j}_{\uparrow\uparrow}$	$r^{i,j}_p$	$r^{i,j}_p + \tau_0$	$r^{i,j}_p - \tau_0$	$r^{i,j}_p$
$\rho^{i,j}_{\uparrow\downarrow}$	$r^{i,j}_p - \frac{\tau_0}{2}$	$r^{i,j}_p + \frac{\tau_0}{2}$	$r^{i,j}_p - \frac{\tau_0}{2}$	$r^{i,j}_p + \frac{\tau_0}{2}$
$\rho^{i,j}_{\downarrow\uparrow}$	$r^{i,j}_p + \frac{\tau_0}{2}$	$r^{i,j}_p + \frac{\tau_0}{2}$	$r^{i,j}_p - \frac{\tau_0}{2}$	$r^{i,j}_p - \frac{\tau_0}{2}$
$\rho^{i,j}_{\downarrow\downarrow}$	$r^{i,j}_p$	$r^{i,j}_p$	$r^{i,j}_p$	$r^{i,j}_p$

Table 3.1: Upper bound on critical delays of paths in process a p.

Example 3.1. *Consider the HSE in Example 2.5. This circuit is called a precharged half buffer(PCHB). There are no initial communications on either port of the PCHB. The repetitive part of its pseudo-repetitive ER system is shown below, along with the collapsed constraint graph. Typical values of the delays for each rule are shown.*

$$E' = \{lo\uparrow, lo\downarrow, li\uparrow, li\downarrow, ri\uparrow, ri\downarrow, ro\uparrow, ro\downarrow\}$$

$$\begin{aligned} R'_1 = \{ & \langle lo\downarrow, i-0\rangle \xrightarrow{4} \langle ro\downarrow, i\rangle \\ & \langle ri\downarrow, i-0\rangle \xrightarrow{2} \langle ro\downarrow, i\rangle \\ & \langle li\downarrow, i-0\rangle \xrightarrow{5} \langle lo\uparrow, i\rangle \\ & \langle ro\downarrow, i-0\rangle \xrightarrow{3} \langle lo\uparrow, i\rangle \\ & \langle lo\uparrow, i-0\rangle \xrightarrow{4} \langle ro\uparrow, i\rangle \\ & \langle li\uparrow, i-0\rangle \xrightarrow{2} \langle ro\uparrow, i\rangle \\ & \langle ri\uparrow, i-0\rangle \xrightarrow{2} \langle ro\uparrow, i\rangle \\ & \langle ro\uparrow, i-1\rangle \xrightarrow{3} \langle lo\downarrow, i\rangle \\ & \} \end{aligned}$$

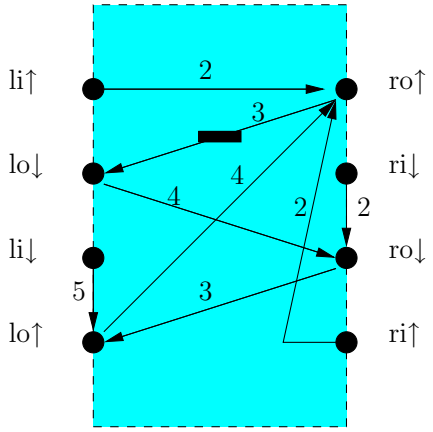


Figure 3.2: Collapsed constraint graph of a PCHB

Note that in the collapsed constraint graph in Figure 3.2, there is a directed cycle

$ro\uparrow \rightarrow lo\downarrow \rightarrow ro\downarrow \rightarrow lo\uparrow \rightarrow ro\uparrow$. The sum of the delays along this cycle is 14. The sum of occurrence index offsets on this cycle is 1. Thus, the cycle time of a circuit containing this process is at least 14. Slack matching only adds buffers to the circuit, and thus does not have any effect on this cycle. Therefore, the target cycle time must be at least 14.

The $\phi_{\uparrow\uparrow}$ path in the collapsed constraint graph consists of the edge $(li\uparrow, ro\uparrow)$. Since the occurrence index offset of this edge is 0, it has critical delay 2 and the $\phi_{\uparrow\uparrow}$ path satisfies the constraint in Table 3.1 for $f \geq 2$.

The $\phi_{\downarrow\downarrow}$ path in the collapsed constraint graph is $li\downarrow \rightarrow lo\uparrow \rightarrow ro\uparrow \rightarrow lo\downarrow \rightarrow ro\downarrow$. The delay of this path is 16. Since the sum of occurrence index offsets of this path is 1, it has critical delay 2 at $\tau_0 = 14$ and the $\phi_{\downarrow\downarrow}$ path satisfies the constraint in Table 3.1 for $f \geq 2$ and $\tau_0 = 14$.

The $\phi_{\uparrow\downarrow}$ path in the collapsed constraint graph is $li\uparrow \rightarrow ro\uparrow \rightarrow lo\downarrow \rightarrow ro\downarrow$. The delay of this path is 9. Since the sum of currency index offsets of this path is 1, it has critical delay -5 at $\tau_0 = 14$ and the $\phi_{\uparrow\downarrow}$ path satisfies the constraint in Table 3.1 for $f \geq 2$ and $\tau_0 = 14$.

The $\phi_{\downarrow\uparrow}$ path in the collapsed constraint graph is $li\downarrow \rightarrow lo\uparrow \rightarrow ro\uparrow$. The delay of this path is 9. Since the sum of occurrence index offsets of the path is 0, it has critical delay 9 and this path satisfies the constrain in Table 3.1 for $f \geq 2$.

Thus, for this particular process, at $\tau_0 = 14$, $f = 2$. Similarly it can be seen that $b = 5$, $l = 5$ and $r = 2$.

Lines [12] analyzes several low latency reshufflings of an LR-buffer. Of these, the following five are half-buffers, where variables Re, Rd implement an output port and variables Le, Ld implement an input port.

$PCHB \equiv *[[Re \wedge Ld]; Rd\uparrow; Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow]$

$WCHB \equiv *[[Re \wedge Ld]; Rd\uparrow; Le\downarrow; [\neg Re \wedge \neg Ld]; Rd\downarrow; Le\uparrow]$

$B1 \equiv *[[Re \wedge Ld]; Rd\uparrow; Le\downarrow; [\neg Re \wedge \neg Ld]; Le\uparrow; Rd\downarrow]$

$B4 \equiv *[[Re \wedge Ld]; Rd\uparrow; Le\downarrow; [\neg Re]; Rd\downarrow, ([\neg Ld]; Le\uparrow)]$

$B5 \equiv *[[Re \wedge Ld]; Rd\uparrow; Le\downarrow; [\neg Re \wedge \neg Ld]; Rd\downarrow, Le\uparrow]$

Consider a buffer such that the projection of the process onto the variables implementing a pair of input and output channels is the PCHB reshuffling with one of the prefixes described below.

- If there is an initial send on the output port, the prefix is

$$[Re]; Rd\uparrow; Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow$$

- If there is an initial receive on the input port, the prefix is

$$[\neg Re]; Rd\downarrow; Le\uparrow$$

- If there are no initial actions on either port, the prefix is

$$Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow$$

A class of buffers that satisfies assumptions 3.1–3.5 was used in the implementation of the Lutonium [19] and MiniMIPS [18].

Example 3.2. *Figure 3.3 shows the collapsed constraint graph of the repetitive part of the pseudo-repetitive ER system that describes the circuit with HSE:*

$$Q \equiv Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow;$$
$$\quad *[[Re \wedge Ld]; Rd\uparrow; Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow]$$
$$S \equiv [Re]; Rd\uparrow; Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow;$$
$$\quad *[[Re \wedge Ld]; Rd\uparrow; Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow]$$
$$R \equiv [\neg Re]; Rd\downarrow; Le\uparrow;$$
$$\quad *[[Re \wedge Ld]; Rd\uparrow; Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow]$$

Typical values of the delays are assigned to each repeated rule. In a manner similar to that shown in Example 3.1 it can be seen that this circuit satisfies assumptions 3.1–3.5 for any cycle time $\tau_0 \geq 14$ *with the following values of the constants in Table 3.1.*

- $f_Q^{R,S} = f_S^{Q,R} = f_R^{S,Q} = 2$
- $b_Q^{S,R} = b_S^{R,Q} = b_R^{Q,S} = 5$

- $l_Q^{R,R} = l_S^{Q,Q} = l_R^{S,S} = 5$
- $r_Q^{S,S} = r_S^{R,R} = r_R^{Q,Q} = 2$

All edges have an occurrence index offset of zero unless they are marked with a rectangular box which denotes an occurrence index offset of 1. If an edge has non-zero delay, it is marked with its delay. Figure 3.4 shows the collapsed constraint graph of

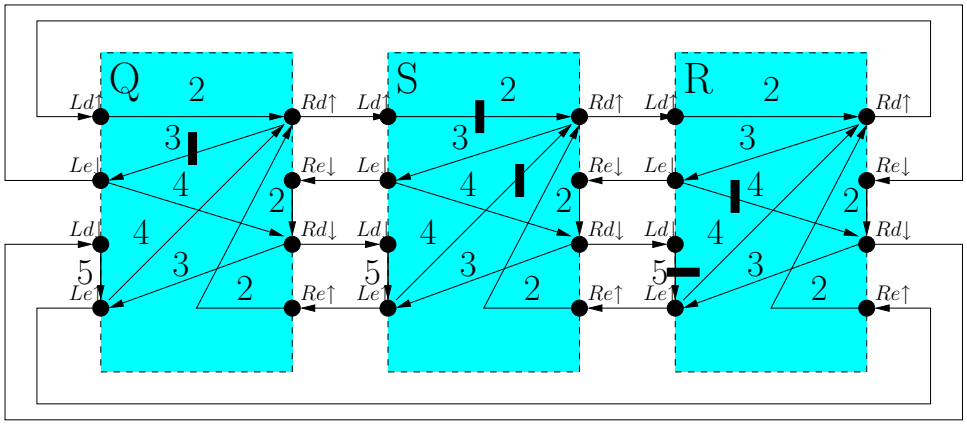


Figure 3.3: Collapsed constraint graph of a ring of buffers satisfying assumptions 3.1–3.5.

the same circuit, however each edge is marked with its critical delay at 14, instead of its delay and occurrence index offset.

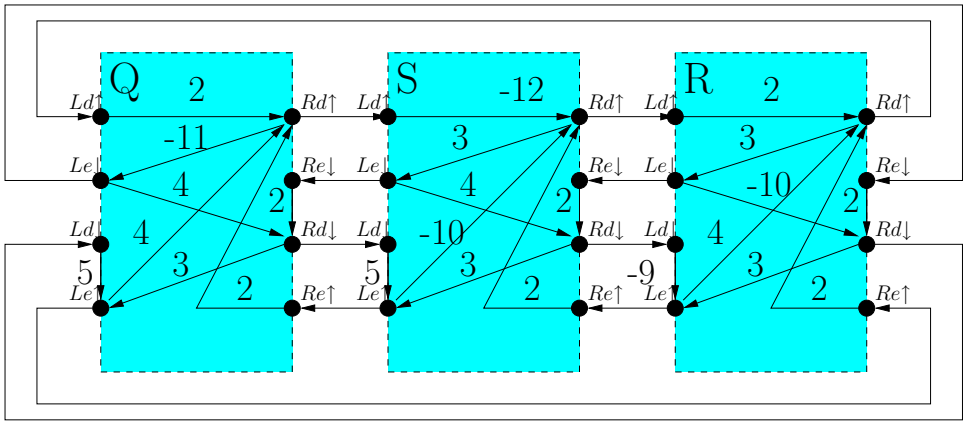


Figure 3.4: Collapsed constraint graph of a ring of buffers with critical delays at 14.

3.1.3 Critical Cycles in the Collapsed Constraint Graph of Circuits of Half Buffers

In this section the critical delay at τ_0 of all cycles in the collapsed constraint graph of a circuit satisfying assumptions 3.1–3.5 is determined.

First in Section 3.1.3.1 a set of regular expressions to characterize all the cycles in such a circuit's collapsed constraint graph is derived. The remainder of this section leads up to Section 3.1.3.4, in which the critical delay of cycles in the collapsed constraint graph of a circuit is determined. In Section 3.1.3.2 the critical delays of certain paths in such a circuit's collapsed constraint graph, assuming there are no initial communications, is determined. In Section 3.1.3.3, these results are used to to determine the critical delay of all cycles in a circuit's collapsed constraint graph, assuming there are no initial communications. Finally, in Section 3.1.3.4 the critical delay of all cycles in the collapsed constraint graph of a circuit with initial communications is determined from the results of Section 3.1.3.3.

3.1.3.1 Cycles in Collapsed Constraint Graphs

The following regular expression characterizes all the cycles in a circuit's collapsed constraint graph. The regular expression is a generalization of those derived by Wong [27]. Wong's regular expressions characterize cycles in circuits comprised only of PCHBs, whereas this regular expression characterizes all cycles in a circuit of buffers that communicate using the four phase handshake.

Lemma 3.1. *All cycles that are not entirely within a process in the collapsed constraint graph of circuit in which all communication channel implement a four phase handshake must match regular expression* (3.4)*:*

$$\mathcal{F} = \phi^* \tag{3.1}$$

$$\mathcal{B} = \beta^* \tag{3.2}$$

$$\mathcal{L} = \mathcal{F}\lambda\mathcal{B}|\mathcal{L}\rho\mathcal{L} \tag{3.3}$$

$$\mathcal{C} = \mathcal{L}\rho|\mathcal{F}|\mathcal{B} \tag{3.4}$$

Proof. Recall that a channel consists of a pair of input and output ports that are connected. For each channel, the input(output) variable of the input port is the output(input) variable of the output port. Let V be the set of variables in a circuit's collapsed constraint graph that implement a port. The set V can be partitioned into two non-intersecting sets V_1 and V_2 such that every $v \in V_1$ is the input variable of an input port and every $v \in V_2$ is the output variable of an input port.

Let a cycle, c, in the collapsed constraint graph of a process be represented as a sequence of variables $\{c_i\}, c_i \in V$. If a cycle traverses no transitions on variables in V, the cycle is entirely within a process. Let $c = \{c_i\}$ be a cycle in the collapsed constraint graph of such a circuit. By definition, if for all i, $c_i \in V_1$, c, must contain only ρ paths. If for all i, $c_i \in V_2$, the cycle must contain only β paths.

If there exists a pair of variables, $c_i, c_j \in c$ such that either $c_i \in V_1, c_j \in V_2$ or $c_i \in V_2, c_j \in V_1$, and either $j = i + 1$ or $j = 1 \wedge i = \|c\|$, then there must exist an even number of such pairs since c is a cycle. For each such pair c_i, c_j, if $c_i \in V_1, c_j \in V_2$ then c contains a λ path. Similarly for such $c_i \in V_2, c_j \in V_1$, c contains a ρ path. Thus such a cycle must contain an equal number of λ and ρ paths, 0 or more ϕ paths between a ρ path and the subsequent λ path and 0 or more β paths between a λ path and the subsequent ρ path. This is exactly the set of cycles described by the regular expression. □

3.1.3.2 Critical Delay of $\mathcal{F}, \mathcal{B}$ and $\mathcal{L}$ Paths

Next, the critical delay of paths matching regular expressions $\mathcal{F}, \mathcal{B}$ and $\mathcal{L}$ in the collapsed constraint graph of a circuit with no initial communications is derived. In

the rest of this section, the variables π, π_1 and π_2 are used to refer to paths in a collapsed constraint graph. i and j represent ports of process p. Similarly h and k represent ports of process q.

Claim 3.1. *In the collapsed constraint graph of a circuit with no initial communications, satisfying assumptions 3.1–3.5, the critical delays at τ_0 of paths traversing only ϕ paths are at most:*

- $\Phi_{\uparrow\uparrow}$: $\sum_{\phi^{i,j}(p) \in \Phi_{\uparrow\uparrow}} f_p^{i,j}$
- $\Phi_{\uparrow\downarrow}$: $\sum_{\phi^{i,j}(p) \in \Phi_{\uparrow\downarrow}} f_p^{i,j} - z \cdot \frac{\tau_0}{2}$
- $\Phi_{\downarrow\uparrow}$: $\sum_{\phi^{i,j}(p) \in \Phi_{\downarrow\uparrow}} f_p^{i,j} + z \cdot \frac{\tau_0}{2}$
- $\Phi_{\downarrow\downarrow}$: $\sum_{\phi^{i,j}(p) \in \Phi_{\downarrow\downarrow}} f_p^{i,j}$

where $z = 1$ if the Φ path contains 1 or more ϕ paths, undefined otherwise.

Proof. The claim is proven via induction. If the path contains no ϕ paths, it is simply a vertex in the collapsed constraint graph. The $\Phi_{\uparrow\uparrow}$ and $\Phi_{\downarrow\downarrow}$ paths clearly have critical delay of 0. There are no $\Phi_{\uparrow\downarrow}$ and $\Phi_{\downarrow\uparrow}$ paths traversing zero ϕ paths since the endpoints of the path are transitions in opposite directions.

The base case of the induction is a Φ path of length 1. Inspection of Table 3.1 easily verifies that the claim holds.

Assume that the claim holds for all Φ paths traversing exactly $n > 1$ ϕ paths. It can be proven that the claim holds for Φ paths traversing $n+1$ ϕ paths by considering the cases where the path of length $n+1$ is a $\Phi_{\uparrow\uparrow}$, a $\Phi_{\uparrow\downarrow}$, a $\Phi_{\downarrow\uparrow}$ and a $\Phi_{\downarrow\downarrow}$ path separately. A Φ path traversing $n+1$ processes must consist of a Φ path traversing n processes, π_1, followed by a path $\pi_2 \in \phi^{h,k}(q)$.

Consider the $\Phi_{\uparrow\uparrow}$ case. Either π_1 is a $\Phi_{\uparrow\uparrow}$ path and π_2 is a $\phi_{\uparrow\uparrow}^{h,k}(q)$ path or π_1 is a $\Phi_{\uparrow\downarrow}$ path and π_2 is a $\phi_{\downarrow\uparrow}^{h,k}(q)$ path. Since π_1 is assumed to traverse at least 1 process, 1 can be substituted for the value of z. In either case, the sum of the critical delays

of the path traversing $n+1$ processes is at most

$$\sum_{\phi^{i,j}(p)\in\Phi_{\uparrow\uparrow}} f_p^{i,j} = \sum_{\phi^{i,j}(p)\in\pi_i} f_p^{i,j} + f_q^{h,k},$$

which proves the claim for $\Phi_{\uparrow\uparrow}$ paths. A similar case analysis can be used to prove the claim for the remaining cases.

If the critical delays of all paths between every process's input and output variables are equal to the values in Table 3.1, the claim specifies not just an upper bound, but the value of the critical delay of all such Φ paths. □

Claim 3.2. *In the collapsed constraint graph of a circuit with no initial communications satisfying assumptions 3.1–3.5, the critical delays at τ_0 of paths traversing only β paths are at most:*

- $B_{\uparrow\uparrow}$: $\sum_{\beta^{i,j}(p)\in B_{\uparrow\uparrow}} b_p^{i,j} - n\frac{\tau_0}{2}$
- $B_{\uparrow\downarrow}$: $\sum_{\beta^{i,j}(p)\in B_{\uparrow\downarrow}} b_p^{i,j} - (n+z)\frac{\tau_0}{2}$
- $B_{\downarrow\uparrow}$: $\sum_{\beta^{i,j}(p)\in B_{\downarrow\uparrow}} b_p^{i,j} - (n-z)\frac{\tau_0}{2}$
- $B_{\downarrow\downarrow}$: $\sum_{\beta^{i,j}(p)\in B_{\downarrow\downarrow}} b_p^{i,j} - n\frac{\tau_0}{2}$

where n is the number of processes traversed and $z = 1$ if $n \geq 1$ undefined otherwise.

Proof. The proof of this claim uses the same techniques as that of claim 3.1. Instead of considering ϕ paths, β paths are considered.

If the path traverses no β paths, it is simply a vertex in the collapsed constraint graph. There are $B_{\uparrow\uparrow}$ and $B_{\downarrow\downarrow}$ paths traversing zero β paths, and their critical delay is 0. There are no $B_{\uparrow\downarrow}$ and $B_{\downarrow\uparrow}$ paths traversing zero β paths since the endpoints of such paths are transitions in opposite directions.

Inspection of Table 3.1 shows that the claim holds for $n = 1$ as well.

Assume towards induction that the claim holds for some $n = n_0 > 1$. Since $n_0 > 1$, the value 1 is substituted for z.

Again, consider the four cases, $B_{\uparrow\uparrow}, B_{\uparrow\downarrow}, B_{\downarrow\uparrow}, B_{\downarrow\downarrow}$, separately. For the $B_{\uparrow\uparrow}$ case, there are two possibilities. Let π_1 be a B path traversing n_0 processes and let π_2 be a $\beta^{h,k}(q)$ path. Either π_1 is a $B_{\uparrow\uparrow}$ path and π_2 is a $\beta^{h,k}_{\uparrow\uparrow}(q)$ path or π_1 is a $B_{\uparrow\downarrow}$ path and π_2 is a $\beta^{h,k}_{\downarrow\uparrow}(q)$ path. The critical delay of $B_{\uparrow\uparrow}$ in the former case is given by

$$\sum_{\beta^{i,j}(p)\in B_{\uparrow\uparrow}} b_p^{i,j} - n_1\frac{\tau_0}{2} = \sum_{\beta^{i,j}(p)\in \pi_1} b_p^{i,j} - n_0\frac{\tau_0}{2} + \beta^{h,k}_{\uparrow\uparrow}(q) - \frac{\tau_0}{2}.$$

Similarly, the critical delay of the path in the latter case is

$$\sum_{\beta^{i,j}(p)\in B_{\uparrow\uparrow}} b_p^{i,j} - n_1\frac{\tau_0}{2} = \sum_{\beta^{i,j}(p)\in \pi_1} b_p^{i,j} - (n_0+1)\frac{\tau_0}{2} + \beta^{h,k}_{\uparrow\uparrow}(q).$$

This proves the claim, for $B_{\uparrow\uparrow}$ paths. A similar analysis for the $B_{\uparrow\downarrow}, B_{\downarrow\uparrow}$ and $B_{\downarrow\downarrow}$ paths completes the proof.

If the critical delays of all paths between every process's input and output variables are equal to the values in Table 3.1, the claim specifies not just an upper bound, but the value of the critical delay of all such B paths. □

Claim 3.3. *Consider any Λ path in the collapsed constraint graph of a circuit with no initial communications satisfying assumptions 3.1–3.5. The critical delay at τ_0 of the path is at most*

- $\Lambda_{\uparrow\uparrow}$: $\sum_{\beta^{i,j}(p)\in\Lambda_{\uparrow\uparrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda_{\uparrow\uparrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda_{\uparrow\uparrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda_{\uparrow\uparrow}} l_p^{i,j} - n\frac{\tau_0}{2}$.
- $\Lambda_{\uparrow\downarrow}$: $\sum_{\beta^{i,j}(p)\in\Lambda_{\uparrow\downarrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda_{\uparrow\downarrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda_{\uparrow\downarrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda_{\uparrow\downarrow}} l_p^{i,j} - (n+1)\frac{\tau_0}{2}$.
- $\Lambda_{\downarrow\uparrow}$: $\sum_{\beta^{i,j}(p)\in\Lambda_{\downarrow\uparrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda_{\downarrow\uparrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda_{\downarrow\uparrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda_{\downarrow\uparrow}} l_p^{i,j} - (n-1)\frac{\tau_0}{2}$.
- $\Lambda_{\downarrow\downarrow}$: $\sum_{\beta^{i,j}(p)\in\Lambda_{\downarrow\downarrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda_{\downarrow\downarrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda_{\downarrow\downarrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda_{\downarrow\downarrow}} l_p^{i,j} - n\frac{\tau_0}{2}$.

where n is the sum of the number of λ and β paths in L and $n \geq 1$.

Proof. This proof uses the same technique of inductively verifying the claim as was used in the proofs of Claim 3.1 and Claim 3.2.

Every Λ path must traverse at least 1 process, and contain at least 1 λ path. The claim is proven by inducting on the number of λ paths contained in Λ.

Base case: A Λ path containing exactly 1 λ path must be a concatenation of Φ, λ and B paths. There are four such possibilities for a $\Lambda_{\uparrow\uparrow}$ path:

1. $\Phi_{\uparrow\uparrow}\lambda_{\uparrow\uparrow}B_{\uparrow\uparrow}$
2. $\Phi_{\uparrow\uparrow}\lambda_{\uparrow\downarrow}B_{\downarrow\uparrow}$
3. $\Phi_{\uparrow\downarrow}\lambda_{\downarrow\uparrow}B_{\uparrow\uparrow}$
4. $\Phi_{\uparrow\downarrow}\lambda_{\downarrow\downarrow}B_{\downarrow\uparrow}$

where Φ paths contain only ϕ paths of a process, and B paths only contain β paths of a process. Let z_Φ be 1 if the Φ path has at least 1 ϕ path, undefined otherwise. Let n_B be the number of β paths in B. Let $z_B = 1$ if $n_B \geq 1$, undefined otherwise. Let n_Λ be the number of β and λ paths in $\Lambda_{\uparrow\uparrow}$.

Note that if z_Φ is undefined paths 3 and 4 do not exist. If z_B is undefined, paths 2 and 4 do not exist. However, path 1 always exists, thus there is always a $\Lambda_{\uparrow\uparrow}$ path.

The critical delays of the paths are respectively at most

1. $\sum_{\beta^{i,j}(p)\in B_{\uparrow\uparrow}} b_p^{i,j} - n_B\frac{\tau_0}{2} + \sum_{\phi^{i,j}(p)\in\Phi_{\uparrow\uparrow}} f_p^{i,j} + l - \frac{\tau_0}{2} = l + \sum_{\beta^{i,j}(p)\in B_{\uparrow\uparrow}} b_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Phi_{\uparrow\uparrow}} f_p^{i,j} - (n_B+1)\frac{\tau_0}{2}$
2. $\sum_{\beta^{i,j}(p)\in B_{\downarrow\uparrow}} b_p^{i,j} - (n_B - z_B)\frac{\tau_0}{2} + \sum_{\phi^{i,j}(p)\in\Phi_{\uparrow\uparrow}} f_p^{i,j} + l - \tau_0 = l + \sum_{\beta^{i,j}(p)\in B_{\uparrow\uparrow}} b_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Phi_{\uparrow\uparrow}} f_p^{i,j} - (n_B - z_B + 2)\frac{\tau_0}{2}$
3. $\sum_{\beta^{i,j}(p)\in B_{\uparrow\uparrow}} b_p^{i,j} - n_B\frac{\tau_0}{2} + \sum_{\phi^{i,j}(p)\in\Phi_{\uparrow\downarrow}} f_p^{i,j} - z_\Phi\frac{\tau_0}{2} + l = l + \sum_{\beta^{i,j}(p)\in B_{\uparrow\uparrow}} b_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Phi_{\uparrow\uparrow}} f_p^{i,j} - (n_B + z_\Phi)\frac{\tau_0}{2}$
4. $\sum_{\beta^{i,j}(p)\in B_{\downarrow\uparrow}} b_p^{i,j} - (n_B - z_B)\frac{\tau_0}{2} + \sum_{\phi^{i,j}(p)\in\Phi_{\uparrow\downarrow}} f_p^{i,j} - z_\Phi\frac{\tau_0}{2} + l - \frac{\tau_0}{2} = l + \sum_{\beta^{i,j}(p)\in B_{\uparrow\uparrow}} b_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Phi_{\uparrow\uparrow}} f_p^{i,j} - (n_B + 1 - z_B + z_\Phi)\frac{\tau_0}{2}$

Note that the critical delay of path 1 clearly satisfies the claim. When z_Φ is defined, the critical delay of 3 also satisfies the claim. When z_B is defined, the critical

delay of 2 satisfies the claim. When z_B and z_Φ are defined, all four paths satisfy the claim. Note that there is always at least one $\Lambda_{\uparrow\uparrow}$ path regardless of the values of z_B and z_Φ.

By considering the remaining three cases in a similar manner, it can be shown that when there is exactly one λ path in a Λ path, the claim holds.

Inductive step: Assume towards induction that the claim holds for all Λ paths containing m_0 λ paths. Let $\Lambda^0_{\uparrow\uparrow}$ be a $\Lambda_{\uparrow\uparrow}$ path with $m = m_0 + 1$ λ paths. This path can be expressed as a composition of a Λ path containing m_0 λ paths, a ρ path and a Λ path containing exactly one λ path. There are four such possibilities:

1. $\Lambda^1_{\uparrow\uparrow}\rho_{\uparrow\uparrow}\Lambda^2_{\uparrow\uparrow}$
2. $\Lambda^1_{\uparrow\uparrow}\rho_{\uparrow\downarrow}\Lambda^2_{\downarrow\uparrow}$
3. $\Lambda^1_{\uparrow\downarrow}\rho_{\downarrow\uparrow}\Lambda^2_{\uparrow\uparrow}$
4. $\Lambda^1_{\uparrow\downarrow}\rho_{\downarrow\downarrow}\Lambda^2_{\downarrow\uparrow}$

where the Λ^1 and Λ^2 paths contain m_0 and one λ paths respectively.

Let n_1 be the number of λ and β paths in Λ^1. Similarly let n_2 be the number of λ and β paths in Λ^2. The number of λ and β paths in $\Lambda^0_{\uparrow\uparrow}$ is $n_0 = n_1 + n_2$.

The critical delays of the paths 1–4 are respectively at most:

1. $\sum_{\beta^{i,j}(p)\in\Lambda^1_{\uparrow\uparrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda^1_{\uparrow\uparrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda^1_{\uparrow\uparrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda^1_{\uparrow\uparrow}} l_p^{i,j} - n_1\frac{\tau_0}{2} + r + \sum_{\beta^{i,j}(p)\in\Lambda^2_{\uparrow\uparrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda^2_{\uparrow\uparrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda^2_{\uparrow\uparrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda^2_{\uparrow\uparrow}} l_p^{i,j} - n_2\frac{\tau_0}{2}$
2. $\sum_{\beta^{i,j}(p)\in\Lambda^1_{\uparrow\uparrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda^1_{\uparrow\uparrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda^1_{\uparrow\uparrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda^1_{\uparrow\uparrow}} l_p^{i,j} - n_1\frac{\tau_0}{2} + r - \frac{\tau_0}{2} + \sum_{\beta^{i,j}(p)\in\Lambda^2_{\downarrow\uparrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda^2_{\downarrow\uparrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda^2_{\downarrow\uparrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda^2_{\downarrow\uparrow}} l_p^{i,j} - (n_2-1)\frac{\tau_0}{2}$
3. $\sum_{\beta^{i,j}(p)\in\Lambda^1_{\uparrow\downarrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda^1_{\uparrow\downarrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda^1_{\uparrow\downarrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda^1_{\uparrow\downarrow}} l_p^{i,j} - (n_1+1)\frac{\tau_0}{2} + r + \frac{\tau_0}{2} + \sum_{\beta^{i,j}(p)\in\Lambda^2_{\uparrow\uparrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda^2_{\uparrow\uparrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda^2_{\uparrow\uparrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda^2_{\uparrow\uparrow}} l_p^{i,j} - n_2\frac{\tau_0}{2}$

4. $\sum_{\beta^{i,j}(p)\in\Lambda^1_{\uparrow\downarrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda^1_{\uparrow\downarrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda^1_{\uparrow\downarrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda^1_{\uparrow\downarrow}} l_p^{i,j} - (n_1+1)\frac{\tau_0}{2} + r + \sum_{\beta^{i,j}(p)\in\Lambda^2_{\downarrow\uparrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda^2_{\downarrow\uparrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda^2_{\downarrow\uparrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda^2_{\downarrow\uparrow}} l_p^{i,j} - (n_2-1)\frac{\tau_0}{2}$

In all four cases, the expressions simplify to

$$\sum_{\beta^{i,j}(p)\in\Lambda^0_{\uparrow\uparrow}} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in\Lambda^0_{\uparrow\uparrow}} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in\Lambda^0_{\uparrow\uparrow}} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in\Lambda^0_{\uparrow\uparrow}} l_p^{i,j} - n_0\frac{\tau_0}{2}$$

This proves the claim for $\Lambda_{\uparrow\uparrow}$ paths with m_0+1 λ paths. A similar case analysis will prove the claim for the three remaining cases of Λ paths completing the induction.

If the critical delays of all paths between every process's input and output variables are equal to the values in Table 3.1, the claim specifies not just an upper bound, but the value of the critical delay of all such Λ paths. □

3.1.3.3 Critical Delay of Cycles in a Circuit with No Initial Communications

In this section, critical delays of paths matching regular expressions $\mathcal{F}, \mathcal{B}$ and $\mathcal{L}$ in the collapsed constraint graph of a circuit with no initial communications are used to determine the critical delay of any cycle in the collapsed constraint graph of circuit with no initial communications.

Lemma 3.2. *A simple cycle, c, in the collapsed constraint graph of a circuit satisfying assumptions 3.1–3.5, with no initial communications has critical delay at most*

$$\sum_{\beta^{i,j}(p)\in c} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in c} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in c} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in c} l_p^{i,j} - n\frac{\tau_0}{2}$$

where n is the number of β and λ paths in the cycle.

Proof. Any cycle in the collapsed constraint graph must match the regular expression in lemma 3.1. If a cycle in the collapsed constraint graph of such a circuit contains no λ edges, the cycle is composed of either only ϕ paths or only β paths. If the cycle is composed of only ϕ paths, for any process p and pair of ports i and j of p, a $\phi^{i,j}(p)$

path occurs in c at most twice, since there are only two transitions that can be the source of a $\phi^{i,j}(p)$ path. Thus c can be written as one of the following:

1. $c = \Phi_{\uparrow\uparrow}$ where $\Phi_{\uparrow\uparrow}$ contains at most one element of $\phi^{i,j}(p)$ for any i, j, p.
2. $c = \Phi_{\downarrow\downarrow}$ where $\Phi_{\downarrow\downarrow}$ contains at most one element of $\phi^{i,j}(p)$ for any i, j, p.
3. $c = \Phi_{\uparrow\downarrow}\Phi_{\downarrow\uparrow}$ where $\Phi_{\uparrow\downarrow}$ and $\Phi_{\downarrow\uparrow}$ each contain at most one element of $\phi^{i,j}(p)$ for any i, j, p.

By claim 3.1, the critical delay of c must be at most $\sum_{\phi^{i,j}(p)\in c} f_p^{i,j}$. Similarly, if c contains only β paths, it can be written as

1. $c = B_{\uparrow\uparrow}$ where $B_{\uparrow\uparrow}$ contains at most one element of $\beta^{i,j}(p)$ for any i, j, p.
2. $c = B_{\downarrow\downarrow}$ where $B_{\downarrow\downarrow}$ contains at most one element of $\beta^{i,j}(p)$ for any i, j, p.
3. $c = B_{\uparrow\downarrow}B_{\downarrow\uparrow}$ where $B_{\uparrow\downarrow}$ and $B_{\downarrow\uparrow}$ each contain at most one element of $\beta^{i,j}(p)$ for any i, j, p.

Let n be the number of β paths in c. Then, by claim 3.2, the critical delay of c is at most $\sum_{\beta^{i,j}(p)\in c} b_p^{i,j} - n\frac{\tau_0}{2}$.

If the cycle contains at least 1 λ path, then c can be written as

1. $\Lambda_{\uparrow\uparrow}\rho_{\uparrow\uparrow}$
2. $\Lambda_{\uparrow\downarrow}\rho_{\downarrow\uparrow}$
3. $\Lambda_{\downarrow\uparrow}\rho_{\uparrow\downarrow}$
4. $\Lambda_{\downarrow\downarrow}\rho_{\downarrow\downarrow}$

Let n be the number of β and λ paths in Λ. By claim 3.3 and assumption 3.4, the critical delay of the c is at most $\sum_{\beta^{i,j}(p)\in c} b_p^{i,j} + \sum_{\rho^{i,j}(p)\in c} r_p^{i,j} + \sum_{\phi^{i,j}(p)\in c} f_p^{i,j} + \sum_{\lambda^{i,j}(p)\in c} l_p^{i,j} - n\frac{\tau_0}{2}$.

If the critical delays of all paths between every process's input and output variables are equal to the values in Table 3.1, the lemma specifies not just an upper bound, but the value of the critical delay of the cycles. □

3.1.3.4 Critical Delay of Cycles in a Collapsed Constraint Graph

In this section, the critical delay of any cycle in the collapsed constraint graph of a circuit with initial communications on a subset of its channels is derived, using the results from Section 3.1.3.3

Theorem 3.1. *Consider a cycle c in the collapsed constraint graph of a circuit satisfying assumptions 3.1–3.5. Let p and q be processes such that there is an initial communication on the channel formed by output port j of p and input port h of q. Let m^+ be the number of pairs of consecutive paths, π_1, π_2, in c such that π_1 is either a $\rho^{i,j}(p)$ or $\phi^{i,j}(p)$ path and π_2 is either a $\lambda^{h,k}(q)$ or $\phi^{h,k}(q)$ path. Let m^- be the number of pairs of consecutive paths π_1, π_2, in c such that π_1 is a $\lambda^{k,h}(q)$ or $\beta^{k,h}(q)$ path and π_2 is a $\rho^{j,i}(p)$ or $\beta^{j,i}(p)$ path. The critical delay of c is at most*

$$\sum_{\beta^{u,v}(w)\in c} b_w^{u,v} + \sum_{\rho^{u,v}(w)\in c} r_w^{u,v} + \sum_{\phi^{u,v}(w)\in c} f_w^{u,v} + \sum_{\lambda^{u,v}(w)\in c} l_w^{u,v} - n\frac{\tau_0}{2} - (m^+ - m^-)\tau_0$$

where n is the number of β and λ paths in c.

Proof. This claim can be proven by induction. Consider a circuit with no initial communications on any channel. The claim is clearly true, by lemma 3.2.

Assume the claim holds for a circuit with n channels with initial communications. The possible pairs of consecutive paths in any cycle of the circuit are $\phi\phi$, $\phi\lambda$, $\rho\phi$, $\rho\lambda$, $\beta\beta$, $\beta\rho$, $\lambda\beta$ and $\lambda\rho$. Consider a pair of consecutive paths $\phi_{\uparrow\uparrow}^{i,j}\phi_{\uparrow\uparrow}^{h,k}$ where the pair of ports j, h are part of a channel with an initial communication, and there are no initial communications on the channels to which ports i and k belong. By Table 3.1, the critical delay of such a path is $f^{i,j} + f^{h,k} - \tau_0$. If there were no initial communication, the delay would be $f^{i,j} + f^{h,k}$. The proof is completed by performing similar comparisons for all possible combinations of consecutive paths, and combinations of initial communications on ports j and h.

If the critical delays of all paths between every process's input and output variables are equal to the values in Table 3.1, the theorem specifies not just an upper bound, but the value of the critical delay of all cycles. □

3.2 Process Graphs and Cycle Time

In this section, process graphs are defined. Sufficient conditions to ensure the circuit's cycle time is at most τ_0 are derived. These conditions are on the process graph.

In Section 3.2.1, process graphs are defined. Next, the relationship between a cycle in the process graph and a cycle in the collapsed constraint graph of a circuit is determined in Section 3.2.2. Then, in Section 3.2.3, a set of sufficient conditions for a circuit to have cycle time at most τ_0 is derived. Finally, in Section 3.2.4, it is shown when these conditions are necessary for the cycle time to be at most τ_0.

3.2.1 Process Graphs

A process graph of a QDI circuit is a directed graph $G = (N, A)$. The set of nodes, N, is the set of processes that comprise the circuit. There is an arc $(u, v) \in A$ if and only if there is a channel comprised of an output port of u and an input port of v. Let an undirected path be a sequence of triplets $\{(u_i, v_i, d_i)\}$ such that

- for all i, $u_i, v_i \in N$, $d_i \in \mathbb{B}$,
- for all i, if $d_i = \textbf{true}$ $(u_i, v_i) \in A$ otherwise $d_i = \textbf{false}$ and $(v_i, u_i) \in A$,
- for all $i > 1$, $u_i = v_{i-1}$.

An undirected path, c, is said to be an undirected cycle if $u_1 = v_{\|c\|}$ where $\|c\|$ is the length of the sequence c.

In the interest of clarity, it is assumed that the process graph does not contain any arcs that are self-loops. In the interest of simplifying the notation, it is also assumed that there is at most one channel (u, v) and at most one channel (v, u) between any pair of processes. It is assumed that there are no arcs $(u, u) \in A$. Whilst the following analysis remains correct without these assumptions, these assumptions allow a simpler notation. I use the convention that input port u of process v is the port that is part of the channel (u, v) and output port w of process v is part of channel (v, w).

Let $D \subseteq N \times N$ be such that $(u, v) \in D$ if and only if there is a *directed* path from u to v in G. Let $U \subseteq N \times N$ be such that $(u, v) \in U$ if and only if there is an *undirected* path from u to v in G. I adopt the convention that $(u, u) \in D \wedge (u, u) \in U$ for all $u \in N$. Define $m_{x,y}$ to be 1 if and only if there is an initial communication on the channel (x, y), 0 otherwise.

3.2.2 Correspondence Between Cycles in Process Graphs and Collapsed Constraint Graphs

There is a direct correspondence between directed cycles in the collapsed constraint graph of a circuit and undirected cycles in its process graph.

Any simple cycle, $c' = \{c'_i\}$ expressed as a sequence of λ,ρ, β and ϕ paths in the collapsed constraint graph of a circuit can be mapped to an undirected cycle, $c = \{c_i\}$ in the circuit's process graph as follows. Let $c'_i = t^{u,v}(w)$.

- If $t \in \{\lambda, \phi\}$, then $c_i = (u, w, \mathbf{true})$.
- If $t \in \{\rho, \beta\}$, then $c_i = (u, w, \mathbf{false})$.

Given an undirected cycle in the process graph, $c = \{(u_i, v_i, d_i)\}$, a corresponding cycle $c' = \{c'_i\}$ in the collapsed constraint graph is constructed as follows. For each pair of triplets $(u_i, v_i, d_i), (u_j, v_j, d_j) \in c$ such that either $j = i + 1$ or $i = \|c\| \wedge j = 1$:

- if $d_i = d_j = \mathbf{true}$, $c'_i \in \phi^{u_i,v_j}(v_i)$.
- if $d_i = d_j = \mathbf{false}$, $c'_i \in \beta^{u_i,v_j}(v_i)$.
- if $d_i = \mathbf{false} \wedge d_j = \mathbf{true}$, $c'_i \in \rho^{u_i,v_j}(v_i)$.
- if $d_i = \mathbf{true} \wedge d_j = \mathbf{false}$, $c'_i \in \lambda^{u_i,v_j}(v_i)$.

with the directions of the transitions chosen such that c' is indeed a cycle in the collapsed constraint graph.

3.2.3 Sufficient Conditions for Slack Matching

Next, a system of linear inequalities that when satisfied imply that a circuit's cycle time is at most τ_0 is derived. Theorem 3.1 provides an upper bound on the critical delay of any cycle in the collapsed constraint graph of a circuit composed of processes satisfying assumptions 3.1–3.5. This upper bound depends only on the number of $t^{i,j}(p)$ paths (for all t, i, j, p) in the cycle. Thus, it suffices to consider the collapsed constraint graphs of such circuits with only $t^{i,j}_{\uparrow\uparrow}(p)$ paths, where i and j are ports of process p and t is as described in Section 3.1.1.

Example 3.3. *Consider the circuit described in example 3.2. Its HSE is:*

$$
\begin{aligned}
Q \equiv\ & Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow; \\
& *[[Re \wedge Ld]; Rd\uparrow; Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow] \\
S \equiv\ & [Re]; Rd\uparrow; Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow; \\
& *[[Re \wedge Ld]; Rd\uparrow; Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow] \\
R \equiv\ & [\neg Re]; Rd\downarrow; Le\uparrow; \\
& *[[Re \wedge Ld]; Rd\uparrow; Le\downarrow; [\neg Re]; Rd\downarrow; [\neg Ld]; Le\uparrow]
\end{aligned}
$$

This circuit satisfies assumptions 3.1–3.5 for any cycle time $\tau_0 \geq 14$ with the following values of the constants in Table 3.1.

- $f_Q^{R,S} = f_S^{Q,R} = f_R^{S,Q} = 2$
- $b_Q^{S,R} = b_S^{R,Q} = b_R^{Q,S} = 5$
- $l_Q^{R,R} = l_S^{Q,Q} = l_R^{S,S} = 5$
- $r_Q^{S,S} = r_S^{R,R} = r_R^{Q,Q} = 2$

Figure 3.5 shows the collapsed constraint graph corresponding to the circuit, with each arc marked by its critical delay at 14. Unmarked arcs have critical delay 0.

Figure 3.6 show the process graph of this circuit. An arc in the process graph is marked with a solid rectangle if there is an initial communication on the channel.

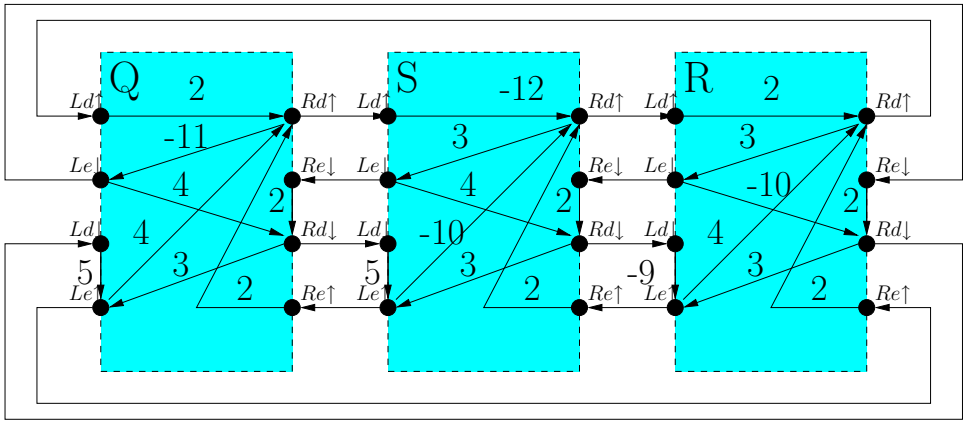


Figure 3.5: Collapsed constraint graph of a ring.

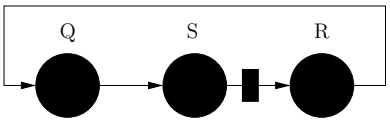


Figure 3.6: Process graph of a ring.

Figure 3.7 shows the collapsed constraint graph corresponding to the circuit, with each arc marked by its critical delay at 14. Unmarked arcs have critical delay 0. Only $t^{i,j}_{\uparrow\uparrow}(p)$ *paths of each process* p *are shown.*

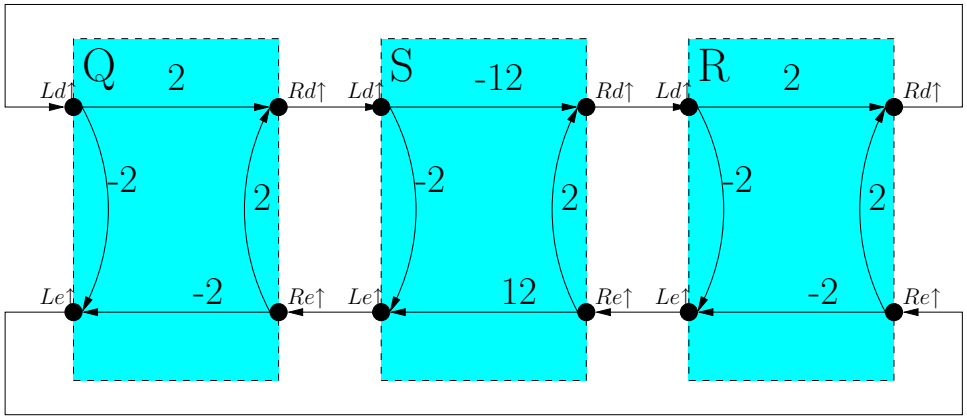


Figure 3.7: Collapsed constraint graph of a ring with only $t^{i,j}_{\uparrow\uparrow}(p)$ paths for each process p.

In the remainder of this section, let $G = (N, A)$ be the process graph of a circuit satisfying assumptions 3.1–3.5. I use the convention that input port u of process v is the port that is part of the channel $(u, v) \in A$ and output port w of process v is part

of channel $(v, w) \in A$. When I refer to the collapsed constraint graph of a circuit, I actually refer to the collapsed constraint graph of the circuit with only $t_{\uparrow\uparrow}^{i,j}(p)$ paths for all $i, j, p \in N$.

In the proceeding, let π', π_1' and π_2' be directed paths in a collapsed constraint graph. The letter c' is used to denote a cycle in a collapsed constraint graph, and the letter c to denote the corresponding cycle in the process graph. The letters $h, i, j, k, s, u, v, w, x, y$ and z are used to identify processes and ports, in both the collapsed constraint graph and the process graph. I assume that each node, k, in the process graph is labeled with the values $f_k^{i,j}, b_k^{i,j}, l_k^{i,j}$ and $r_k^{i,j}$ that satisfy assumption 3.4 for all i, j.

In order to verify that a circuit has cycle time at most τ_0, it must be verified that each directed cycle in its collapsed constraint graph has non-positive critical delay at τ_0. Each directed cycle in the collapsed constraint graph corresponds to an undirected cycle in the process graph. Thus, given a process graph of a circuit, the list of channels with initial communications and the values of $f_k^{i,j}, b_k^{i,j}, l_k^{i,j}$ and $r_k^{i,j}$, for each undirected cycle in the process graph the critical delay of the corresponding directed cycle in the collapsed constraint graph can be computed. There are a possibly exponential number of undirected cycles in a graph. Thus, in the remainder of this section, a polynomial sized set of linear inequalities is derived that is sufficient to guarantee a circuit has cycle time at most τ_0.

Consider $\Phi^{w,z}$ paths in the circuits collapsed constraint graph that traverse only ϕ paths of any process. Let w be an input port of process x and z be an output port of process y. For each set of such nodes in the process graph, let $\mathcal{F}_{wxyz} \cdot \tau_0$ be an upper bound on the critical delay of any $\Phi_{\uparrow\uparrow}^{w,z}$ path from an input variable of port w of process x to an output variable of port z of process y. It will be shown that all

$\mathcal{F}_{wxyz}$ must satisfy the following inequalities.

$$Sf_{xy} \geq -m_{x,y} \qquad \forall x, y \in N : (x, y) \in A \tag{3.5}$$

$$\mathcal{F}_{wxxy} \geq \frac{f_x^{w,y}}{\tau_0} + Sf_{xy} \qquad \forall w, x, y \in N : (w, x), (x, y) \in A \tag{3.6}$$

$$\mathcal{F}_{wxyz} \geq \mathcal{F}_{wxuy} + \mathcal{F}_{uyyz} \quad \forall w, x, u, y, z \in N : (w, x), (u, y), (y, z) \in A \wedge (x, u) \in D \tag{3.7}$$

Claim 3.4. *Let π' be a $\Phi_{\uparrow\uparrow}^{w,z}$ path in a circuit's collapsed constraint graph that traverses only ϕ paths. Furthermore, let w be an input port of process x. Similarly, let z be an output port of process y. Then, satisfying inequalities* (3.5)–(3.7) *implies that $\mathcal{F}_{wxyz} \cdot \tau_0$ is at least as large as the critical delay of π'.*

Proof. The claim is proven via induction. Before doing so, the critical delay of any such π' is established.

The critical delay of any such π' is given by:

$$\sum_{\phi^{h,j}(i) \in \pi'} f_i^{h,j} - m_{i,j} \cdot \tau_0$$

This follows from claim 3.1 if there are no initial communications on any channel in the system. If there are initial communications, recall that the collapsed constraint graph only contains $\phi_{\uparrow\uparrow}$ paths. From assumption 3.4 and Table 3.1, the critical delay of any $\phi_{\uparrow\uparrow}^{h,j}(i)$ path is given by $f_i^{h,j} - m_{i,j}\tau_0$.

Base Case: π' must be of length at least one. Let $\pi' = \phi^{u,w}(v)$. If inequalities (3.5)–(3.7) are satisfied, there exist Sf_{vw} and $\mathcal{F}_{uvvw}$ such that

$$Sf_{vw} \geq -m_{v,w}$$
$$\mathcal{F}_{uvvw} \geq \frac{f_v^{u,w}}{\tau_0} + Sf_{vw}$$

Substituting the former inequality into the latter proves the claim.

Inductive Step: Assume that the claim holds for all π' of length n. Consider a path, π', of length $n+1$. This path can be expressed as a composition of a path, π'_1 of length n and a path π'_2 of length 1.

Let π'_1 be a $\Phi^{u,w}$ path and π'_2 be a $\phi^{y,x}(w)$ path, where $(u,v),(y,w),(w,x) \in A$.

If inequalities (3.5)–(3.7) are satisfied, there exist $\mathcal{F}_{ywwx}, \mathcal{F}_{uvwx}$ and $\mathcal{F}_{uvyw}$ that satisfy the following inequality.

$$\mathcal{F}_{uvwx} \geq \mathcal{F}_{uvyw} + \mathcal{F}_{ywwx}$$

Furthermore, by inductive assumption, $\mathcal{F}_{uvyw}$ is at least the critical delay of π'_1 and from the proof of the base case, $\mathcal{F}_{ywwx}$ is at least the critical delay of π'_2. The critical delay of π' is the sum of the critical delay of π'_1 and π'_2. Thus, $\mathcal{F}_{uvwx}$ is at least as large as the critical delay of π'. □

Thus it has been shown that $\mathcal{F}_{uvwx} \cdot \tau_0$ is at least as large as the critical delay of any $\Phi^{u,x}_{\uparrow\uparrow}$ path that traverses only ϕ paths, where u is an input port of process v and x is an output port w. Satisfying the following inequality ensures that critical delay of any cycle in the collapsed constraint graph traversing only ϕ paths is non-positive.

$$\mathcal{F}_{xyxy} \leq 0 \qquad \forall x, y \in N : (x,y) \in A, (y,x) \in D \tag{3.8}$$

Lemma 3.3. *All cycles in a circuit's collapsed constraint graph that traverse only ϕ paths have critical delay at most* 0*, if inequalities* (3.5)–(3.8) *can be satisfied.*

Proof. As shown in Section 3.2.2, any directed cycle c in G has a corresponding cycle c' in the collapsed constraint graph that traverses only ϕ paths. Similarly, any cycle c' in the collapsed constraint graph traversing only ϕ paths has a corresponding directed cycle c in G. Since there are no arcs $(u,u) \in A$, c must have length at least 2.

Let the arcs (x,y) and (y,z) occur consecutively in c. Thus, there must be a directed path from z to x, and consequently a directed path from y to y containing arcs (y,z) and (x,y).

Let this path from y to y be π, and the corresponding path in the collapsed constraint graph traversing only $\phi_{\uparrow\uparrow}$ paths be π'. By claim 3.4, $\mathcal{F}_{xyxy} \cdot \tau_0$ must be at least as large as the critical delay of π'. However π' is exactly the cycle c'. Thus $\mathcal{F}_{xyxy} \cdot \tau_0$ is at least as large as the critical delay of c'. If inequality (3.8) is satisfied, then c' has critical delay at most 0. □

Next, $B^{w,z}$ paths in the circuit's collapsed constraint graph that traverse only β paths of any process are considered. Let w be an output port of process x and z be an input port of process y. For each set of such nodes in the process graph, let $-\mathcal{B}_{wxyz} \cdot \tau_0$ be an upper bound on the critical delay of any $B^{w,z}_{\uparrow\uparrow}$ path from an input variable of port w of process x to an output variable of port z of process y. It is shown that all $\mathcal{B}_{wxyz}$ must satisfy the following inequalities.

$$Sb_{yx} \leq -m_{x,y} \qquad \forall x, y \in N : (x, y) \in A \tag{3.9}$$

$$\mathcal{B}_{wxxy} \leq \frac{1}{2} - \frac{b_x^{w,y}}{\tau_0} + Sb_{wx} \qquad \forall w, x, y \in N : (y, x), (x, w) \in A \tag{3.10}$$

$$\mathcal{B}_{wxyz} \leq \mathcal{B}_{wxuy} + \mathcal{B}_{uyyz} \qquad \forall w, x, u, y, z \in N : (z, y), (y, u), (x, w) \in A, (u, x) \in D \tag{3.11}$$

Claim 3.5. *Let π' be a $B^{w,z}$ path in a circuit's collapsed constraint graph that traverses only β paths of any process. Further more, let z be an input port of process y. Similarly, let w be an output port of process x. Then, satisfying inequalities* (3.9)–(3.11) *implies that $-\mathcal{B}_{wxyz} \cdot \tau_0$ is an upper bound on the critical delay of π'.*

Proof. The proof of this claim is nearly identical to that of claim 3.4. The only difference is that the delays of B paths is considered, not that of Φ paths. The claim is proven via induction. First, the critical delay of any such π' is established.

The critical delay of any such π' is given by:

$$\sum_{\beta^{h,j}(i) \in \pi'} b_i^{h,j} - \frac{\tau_0}{2} + m_{i,h} \cdot \tau_0$$

This follows from claim 3.2 if there are no initial communications on any channel in the system. If there are initial communications, recall that the collapsed constraint graph only contains $\beta_{\uparrow\uparrow}$ paths. By assumption 3.4 and Table 3.1, the critical delay of any $\beta^{h,j}_{\uparrow\uparrow}(i)$ path is given by $b^{h,j}_i - \frac{\tau_0}{2} + m_{i,h}\tau_0$.

Base Case: π' must be of length at least one. Let $\pi' = \beta^{u,w}(v)$. If inequalities (3.9)–(3.11) are satisfied, there exist Sb_{uv} and $\mathcal{B}_{uvvw}$ such that

$$Sb_{uv} \leq -m_{v,u}$$
$$\mathcal{B}_{uvvw} \leq \frac{1}{2} - \frac{b^{u,w}_v}{\tau_0} + Sb_{uv}$$

Substituting the former inequality into the latter completes the proof.

Inductive Step: Assume that the claim holds for all π' of length n. Consider a path, π', of length $n+1$. This path can be expressed as a composition of a path, π'_1 of length n and a path π'_2 of length 1.

Let π'_1 be a $B^{u,w}$ path and π'_2 be a $\beta^{y,x}(w)$ path, where $(v,u),(w,y),(x,w) \in A$.

If inequalities (3.9)–(3.11) are satisfied, there exist $\mathcal{B}_{ywwx}, \mathcal{B}_{uvwx}$ and $\mathcal{B}_{uvyw}$ that satisfy the following inequality.

$$\mathcal{B}_{uvwx} \leq \mathcal{B}_{uvyw} + \mathcal{B}_{ywwx}$$

Furthermore, by inductive assumption, $-\mathcal{B}_{uvyw}\tau_0$ is at least the critical delay of π'_1 and from the proof of the base case, $-\mathcal{B}_{ywwx}\tau_0$ is at least the critical delay of π'_2. The critical delay of π' is the sum of the critical delay of π'_1 and π'_2. Thus, $-\mathcal{B}_{uvwx}\tau_0$ is at least as large as the critical delay of π'. □

Thus, it has been shown that $-\mathcal{B}_{uvwx} \cdot \tau_0$ is at least as large as the critical delay of any $B^{u,x}_{\uparrow\uparrow}$ path that traverses only β paths, where u is an output port of process v and x is an input port of w. Satisfying the following inequality ensures that the critical delay of any cycle in the collapsed constraint graph traversing only β paths is not positive.

$$\mathcal{B}_{xyxy} \geq 0 \qquad \forall x,y \in N : (y,x) \in A, (x,y) \in D \tag{3.12}$$

Lemma 3.4. *All cycles in a circuit's collapsed constraint graph that traverse only β paths have critical delay at most* 0*, if inequalities* (3.9)–(3.12) *can be satisfied.*

Proof. The proof of this lemma, is nearly identical to that of lemma 3.3. The only difference is that cycles traversing β paths instead of those traversing ϕ paths are considered.

As shown in Section 3.2.2, any undirected cycle c in G has a corresponding cycle c' in the collapsed constraint graph that traverses only β paths. Similarly, any cycle c' in the collapsed constraint graph traversing only β paths has a corresponding undirected cycle c in G. Since there are no arcs $(u, u) \in A$, c must have length at least 2.

Let the triples $(x, y, \textbf{false})$ and $(y, z, \textbf{false})$ occur consecutively in c. Thus, there must be a directed path from x to z, and consequently a directed path from y to y containing arcs (y, z) and (x, y).

Let π be the undirected path obtained by traversing each edge on this path from y to y in the opposite direction. Let the corresponding path in the collapsed constraint graph traversing only $\beta_{\uparrow\uparrow}$ paths be π'.

By claim 3.5, $-\mathcal{B}_{xyxy} \cdot \tau_0$ must be at least as large as the critical delay of π'. However π' is exactly the cycle c'. Thus $-\mathcal{B}_{xyxy} \cdot \tau_0$ is at least as large as the critical delay of c'. If inequality (3.12) is satisfied, then c' has critical delay at most 0. □

I have thus far derived constraints that must be satisfied if the critical delay of any cycle in the collapsed constraint graph that traverses only ϕ or only β paths is non-positive. However, there may be cycles that contain λ and ρ paths. Consider a $\Lambda_{\uparrow\uparrow}^{w,z}$ path in the collapsed constraint graph, such that w is an input port of process x and z and input port of process y. For each set of such nodes in the process graph, let $-\mathcal{L}_{wxyz} \cdot \tau_0$ be an upper bound on the critical delay of any $\Lambda_{\uparrow\uparrow}^{w,z}$ path from an input variable of port w of process x to an output variable of port z of process y. It

is shown that $\mathcal{L}_{wxyz}$ must satisfy the following inequalities.

$$\mathcal{L}_{wxxy} \leq \frac{1}{2} - \frac{l_x^{w,y}}{\tau_0}$$
$$\forall w, x, y \in N : (w, x), (y, x) \in A \tag{3.13}$$

$$\mathcal{L}_{wxyz} \leq \mathcal{L}_{wxuv} + \mathcal{B}_{uvyz}$$
$$\forall u, v, w, x, y, z \in N : (w, x), (v, u), (z, y) \in A, \exists s \in N : (y, v), (x, s), (u, s) \in D \tag{3.14}$$

$$\mathcal{L}_{wxyz} \leq -\mathcal{F}_{wxuv} + \mathcal{L}_{uvyz}$$
$$\forall u, v, w, x, y, z \in N : (w, x), (u, v), (z, y) \in A, \exists s \in N : (x, u), (v, s), (y, s) \in D \tag{3.15}$$

$$\mathcal{L}_{wxyz} \leq \mathcal{L}_{wxuv} - \mathcal{R}_{uvvs} + \mathcal{L}_{vsyz}$$
$$\forall w, x, y, z, u, v, s \in N : (w, x), (v, u), (v, s), (z, y) \in A, (u, x), (y, s) \in U \tag{3.16}$$

$$\mathcal{R}_{wxxy} \geq -Sb_{wx} + \frac{r_x^{w,y}}{\tau_0} + Sf_{xy}$$
$$\forall w, x, y \in N : (x, w), (x, y) \in A \tag{3.17}$$

Claim 3.6. *Let π' be a $\Lambda^{u,x}$ path in a circuit's collapsed constraint graph. Furthermore, let u be an input port of process v. Similarly, let x be an input port of process w. Then, satisfying inequalities* (3.5)–(3.17) *implies that $-\mathcal{L}_{uvwx} \cdot \tau_0$ is an upper bound on the critical delay of π'.*

Proof. This proof is very similar to that of claims 3.4 and 3.5.

The claim is proven via induction on the number of λ paths in π'. Before doing so, the critical delay of any such π' is established. The critical delay of any such π' is given by:

$$\sum_{\substack{\lambda^{h,j}(i) \\ \in \pi'}} l_i^{h,j} - \frac{\tau_0}{2} + \sum_{\substack{\beta^{h,j}(i) \\ \in \pi'}} b_i^{h,j} - \frac{\tau_0}{2} + m_{i,h} \cdot \tau_0 + \sum_{\substack{\phi^{h,j}(i) \\ \in \pi'}} f_i^{h,j} - m_{i,j} \cdot \tau_0 + \sum_{\substack{\rho^{h,j}(i) \\ \in \pi'}} r_i^{h,j} - m_{i,j} \cdot \tau_0 + m_{i,h} \cdot \tau_0$$

This follows from claim 3.1 if there are no initial communications on any channel in the system. If there are initial communications, recall that the collapsed constraint graph only contains $\phi_{\uparrow\uparrow}$ paths. By assumption 3.4 and Table 3.1, the critical delay of any $\phi^{h,j}(i)$ path is given by $f_i^{h,j} - m_{i,j}\tau_0$. Similarly, the critical delay of any $\beta^{h,j}(i)$ path is given by $b_i^{h,j} + m_{i,h} \cdot \tau_0$ and the critical delay of any $\rho^{h,j}(i)$ path is given by $r_i^{h,j} - m_{i,j} \cdot \tau_0 + m_{i,h} \cdot \tau_0$.

π' by definition must contain at least one λ path.

Base case: Let $\pi' = \lambda_{\uparrow\uparrow}^{w,y}(x)$. The claim is proven for π' that contain exactly one λ path by considering three cases:

- $\pi' = \lambda_{\uparrow\uparrow}^{w,y}(x)$,
- $\pi' == \lambda_{\uparrow\uparrow}^{u,y}(v) B_{\uparrow\uparrow}^{v,x}$ where $B_{\uparrow\uparrow}^{v,x}$ traverses one or more β paths, and
- $\pi' = \Phi^{u,z}\Lambda^{y,x}$ such that $\Phi^{u,z}$ contains one or more ϕ paths and $\Lambda^{y,x}$ contains one λ path and zero or more β paths.

If inequalities (3.5)–(3.17) hold, there exists $\mathcal{L}_{wxxy}$ such that

$$\mathcal{L}_{wxxy} \leq \frac{1}{2} - \frac{l_x^{w,y}}{\tau_0}.$$

This proves the claim for $\pi' = \lambda_{\uparrow\uparrow}^{w,y}(x)$.

If $\pi' = \lambda_{\uparrow\uparrow}^{u,y}(v) B_{\uparrow\uparrow}^{v,x}$, where $B^{v,x}$ only has β paths, and $B_{\uparrow\uparrow}^{v,x}$ is a path from port v of process y to port x of process w. Inequalities (3.5)–(3.17) hold so there exist $\mathcal{L}_{uvvy}, \mathcal{B}_{vywx}$ and $\mathcal{L}_{uvwx}$ such that

$$\mathcal{L}_{uvwx} \leq L_{uvvy} + \mathcal{B}_{vywx}.$$

Since inequalities (3.5)–(3.17) hold, claim 3.5 holds. By claim 3.5, $-\mathcal{B}_{vywx} \cdot \tau_0$ is an upper bound on the critical delay of $B_{\uparrow\uparrow}^{v,x}$. This, along with the analysis for $\pi' = \lambda_{\uparrow\uparrow}^{w,y}(x)$ proves the claim for $\pi' = \lambda_{\uparrow\uparrow}^{u,y}(v) B_{\uparrow\uparrow}^{v,x}$.

Lastly, consider the case where $\pi' = \Phi^{u,z}\Lambda^{y,x}$ such that $\Phi^{u,z}$ contains only ϕ paths and $\Lambda^{y,x}$ contains one λ path and zero or more β paths. Furthermore let u be a port

of process v, z a port of process y, y a port of process z, x a port of process w.

It has been established that the claim holds for the $\Lambda^{y,x}$ path. Since inequalities (3.5)–(3.17) hold, claim 3.4 holds. There also exist $\mathcal{F}_{uvyz}, \mathcal{L}_{yzwx}$ and $\mathcal{L}_{uvwx}$ such that

$$\mathcal{L}_{uvwx} \leq -\mathcal{F}_{uvyz} + \Lambda_{yzwx}.$$

The claim follows from this inequality, claim 3.4 and the claim for $\pi' = \lambda^{u,y}_{\uparrow\uparrow}(v)B^{v,x}_{\uparrow\uparrow}$ proven above.

Inductive step: Assume that the claim holds for all paths π' containing n λ paths. Consider a path π' with $n+1$ λ paths. $\pi' = \Lambda^{u,y}\rho^{s,z}(y)\Lambda^{y,x}$ where $\Lambda^{u,y}$ has n λ paths and $\Lambda^{y,x}$ has one λ path. Furthermore, let $(y,s),(y,z),(x,w),(u,v) \in A$. Satisfying inequalities (3.5)–(3.17) implies there exist $\mathcal{L}_{uvsy}, \mathcal{R}_{syyz}, \mathcal{L}_{yzwx}, Sb_{sy}, Sf_{yz}$ and $\mathcal{L}_{uvwx}$ such that

$$\mathcal{R}_{syyz} \geq \frac{r_y^{s,z}}{\tau_0} + Sf_{yz} - Sb_{sy}$$
$$\mathcal{L}_{uvwx} \leq \mathcal{L}_{uvsy} - \mathcal{R}_{syyz} + \mathcal{L}_{yzwx}$$

By the inductive hypothesis and the base case, the claim holds for $\Lambda^{u,y}$ and $\Lambda^{y,x}$. Assumption 3.4 and Table 3.1 show that the critical delay of any $\rho^{s,z}(y)$ is at most $\mathcal{R}_{syyz} \cdot \tau_0$. This along with the last of the two inequalities proves the claim. □

Next, all cycles in a circuit's collapsed constraint graph such that the cycle traverses at least one λ path are considered. The following two constraints ensure that the critical delay of all such cycles is non-positive.

$$\mathcal{C}_{xw} \leq \mathcal{L}_{xwyx} - \mathcal{R}_{yxxw} \qquad \forall w,y,x \in N : (x,w),(x,y) \in A, (w,y) \in U \tag{3.18}$$
$$\mathcal{C}_{xw} \geq 0 \qquad \forall (x,w) \in A \tag{3.19}$$

Theorem 3.2. *All cycles in a circuit's collapsed constraint graph have critical delay at most* 0*, if inequalities* (3.5)–(3.19) *can be satisfied.*

Proof. If a cycle does not contain any λ paths, then by lemmas 3.3 and 3.4 the

theorem is true.

If the cycle contains at least one λ path, then the cycle can be written as $\Lambda^{x,x}\rho_x^{y,w}$. Satisfying inequalities (3.5)–(3.19) implies there exists $\mathcal{L}_{xwyx}, \mathcal{R}_{yxxw}$ and C_{xw} such that

$$\mathcal{C}_{xw} \leq \mathcal{L}_{xwyx} - \mathcal{R}_{yxxw} \qquad \forall w, y, x \in N : (x, w), (x, y) \in A, (w, y) \in U$$
$$\mathcal{C}_{xw} \geq 0 \qquad \forall (x, w) \in A$$

By claim 3.6, $-\mathcal{L}_{xwyx} \cdot \tau_0$ is an upper bound on the critical delay of the $\Lambda^{x,x}$ path in the cycle. Assumption 3.4 shows that $R_{yxxw}\tau_0$ is an upper bound on the critical delay of the $\rho_x^{y,w}$ path in the cycle. Thus, $-\mathcal{C}_{xw} \cdot \tau_0$ is an upper bound on the critical delay of the cycle. The above inequality requires that this be at most 0, proving the claim. □

3.2.4 Necessary Conditions for Slack Matching

Next, it is shown when inequalities (3.5)–(3.19) must be satisfied for a circuit to have cycle time at most τ_0.

Lemma 3.5. *Let p be a process in a circuit with ports i and j. If all $t^{i,j}(p)$ paths have critical delay at τ_0 equal to the values in Table 3.1, then inequalities* (3.5)–(3.19) *are necessary for the circuit to have cycle time τ_0.*

Proof. Let Q be a circuit such that for all processes, every path between any pair of input and output transitions the critical delay at τ_0 equals the value in Table 3.1. Let Q have cycle time less than τ_0. Let $G = (N, A)$ be the process graph of Q. Inequalities (3.5)–(3.19) can be satisfied as follows.

Let inequalities (3.5),(3.6), (3.9),(3.10),(3.13) and (3.17) be satisfied with equality.

Let w, x, y, z and π' be as in claim 3.4. Choose $\mathcal{F}_{w,x,y,z}$ such that the claim holds for all π' and is satisfied with equality for some π'. This is done by choosing $\mathcal{F}_{wxyz}$ such that for all $u, y : (u, y) \in A, (x, u) \in D$ inequality (3.7) holds, and is satisfied with equality for some u and y.

Similarly, let w, x, y, z and π' be as in claim 3.5. Choose $\mathcal{B}_{wxyz}$ such that the claim holds for all π' and is satisfied with equality for some π'. This is done by choosing $\mathcal{B}_{wxyz}$ such that for all $u, y : (y, u) \in A, (u, x) \in D$ inequality (3.11) holds, and is satisfied with equality for some u and y.

For w, x, y, z and π' as in claim 3.6, choose $\mathcal{L}_{wxyz}$ such that the claim holds for all π' and is satisfied with equality for some π'. This is done by choosing $\mathcal{L}_{wxyz}$ so that for all u, v, s if

1. $(v, u) \in A \wedge (y, v), (x, s), (u, s) \in D$, then inequality (3.14) holds for all such (u, v, s),

2. $(u, v) \in A \wedge (y, s), (x, u), (v, s) \in D$, then inequality (3.15) holds for all such (u, v, s),

3. $(v, u), (v, s) \in A \wedge (u, x), (y, s) \in U$, then inequality (3.16) holds for all such (u, v, s)

and for every $\mathcal{L}_{wxyz}$ at least one of the inequalities (3.14)–(3.16) is satisfied with equality.

If one of the inequalities (3.8),(3.12), (3.18)–(3.19) is not satisfied, then by construction there exists some cycle in the process graph, such that critical delay at τ_0 of the corresponding cycle in the collapsed constraint graph is greater than zero. (This follows from claims 3.4–3.6 and theorem 3.1). □

3.3 Slack Matching a QDI circuit

In this section, I derive a mixed integer linear program(MILP) that can be solved to determine the number and placement of slack matching buffers needed for any circuit composed of processes satisfying assumptions 3.1–3.5 to have cycle time τ_0.

First, in Section 3.3.1, results from Section 3.1 and 3.2 are used to establish the critical delays between input and output variables of a pipeline of LR-buffers. Next, in Section 3.3.2, it is shown how this can be used to generate a set of inequalities over

a collection of real and integer variables that must be satisfied in order for the circuit to have cycle time τ_0. Finally, in Section 3.3.3, the time complexity of generating this MILP, given a circuit's process graph, is determined.

3.3.1 Pipelines of LR-buffers

In Section 3.2, I derived a set of constraints sufficient to guarantee that a circuit's cycle time is at most τ_0. Only an integral number of slack matching buffers can be added to any channel. In the following section, such a set of inequalities is constructed for the circuit obtained by adding a variable number of buffers, $n_{u,v}$, to each channel (u, v) in a circuit. In order to do so the critical delays of paths between input and output variables of a pipeline of LR-buffers need to be characterized. As explained in Section 3.2, only the critical delay of any path traversing only $\phi_{\uparrow\uparrow}, \beta_{\uparrow\uparrow}, \lambda_{\uparrow\uparrow}$ and $\rho_{\uparrow\uparrow}$ paths needs to be considered.

Let s be the buffer that is used for slack matching a system. Let i be the input port of s and j the output port. Furthermore, let s satisfy assumptions 3.1–3.5 and let $f_s^{i,j} = r_s^{j,j}$, $b_s^{j,i} = l_s^{i,i}$ and $f_s^{i,j} + b_s j, i \leq \frac{\tau_0}{2}$. An example of such a buffer was shown in Example 3.1. By claim 3.1, the critical delay of any $\Phi_{\uparrow\uparrow}$ path traversing only ϕ paths in a pipeline of n instances of s is given by $n \cdot f_s^{i,j}$. Since only $t_{\uparrow\uparrow}$ paths in each slack matching buffer are being considered, all simple $\Phi_{\uparrow\uparrow}$ paths in the pipeline contain only $\phi_{\uparrow\uparrow}$ paths. Symmetrically, it can be argued that the critical delay of any $B_{\uparrow\uparrow}$ path is at most $n(b_s^{j,i} - \frac{\tau_0}{2})$. All simple $\Lambda_{\uparrow\uparrow}$ paths must contain an equal number of $\phi_{\uparrow\uparrow}$ paths and $\beta_{\uparrow\uparrow}$ paths and one $\lambda_{\uparrow\uparrow}$ path. The sum of the critical delay of a pair of $\phi_{\uparrow\uparrow}$ and $\beta_{\uparrow\uparrow}$ paths is $f_s^{i,j} + b_s^{j,i} - \frac{\tau_0}{2} \leq 0$. Thus, critical $\Lambda_{\uparrow\uparrow}$ path has critical delay $l_s^{i,i}$. A symmetric argument can be used to show that the critical $P_{\uparrow\uparrow}$ path has critical delay $r_s^{j,j}$.

3.3.2 MILP for Slack Matching

In this section, SMOP is stated as a MILP and it is shown how to generate this MILP efficiently. It is assumed that the circuit being slack matched is closed, that is each

port of a process is part of a channel containing the port of another process. If the circuit is not closed, a source is connected to each of the circuit's input ports and a sink to each of the system's output ports. A source is a buffer with exactly one output channel, and a sink is a buffer with exactly one input channel. 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.

Consider a process graph $G = (N, A)$ representing a circuit comprised of processes satisfying assumptions 3.1–3.5. The circuit represented by G has cycle time at most τ_0 if theorem 3.2 holds.

Slack matching is performed by adding buffers along communication channels in such a manner that the inequalities (3.5)–(3.19) *can be satisfied for the resulting system. Slack matching only changes a system by adding buffers to communication channels.*

Replacing inequalities (3.5) *and* (3.9) *by* (3.20)–(3.22)*, the set of inequalities from theorem 3.2 is obtained for the system with n_{uv} slack matching buffers added to each channel (u, v), where the slack matching buffer, s, is as described in the previous section and the following assumption holds.* It is assumed that $l_s^{i,i} \leq l_p^{j,j}$ for all process p and ports j in the circuit being slack matched. Similarly, it is assumed that $r_s^{i,i} \leq r_p^{j,j}$ for all process p and ports j in the circuit being slack matched. This is typically the case for LR-buffers. *Slack matching a circuit is thus equivalent to determining non-negative integers n_{uv} such that the system of inequalities* (3.6)–(3.8) *and* (3.10)–(3.22) *is satisfied.*

$$Sf_{uv} \geq -m_{u,v} + n_{uv}\frac{f_s^{i,j}}{\tau_0} \qquad \forall (u,v) \in A \tag{3.20}$$

$$Sb_{vu} \leq -m_{u,v} + n_{uv} \cdot \left(\frac{1}{2} - \frac{b_s^{i,j}}{\tau_0}\right) \qquad \forall (u,v) \in A \tag{3.21}$$

$$n_{uv} \in \mathbb{N} \qquad \forall (u,v) \in A \tag{3.22}$$

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

3.3.3 Generating the MILP

The time complexity of generating the MILP to slack match a circuit is analyzed in this section. Let n be the number of nodes in the collapsed constraint graph of a circuit. Let m be the number of arcs in the collapsed constraint graph.

Given the collapsed constraint graph, G, a $n \times n$ matrix, $\mathcal{D}$, such that $\mathcal{D}_{u,v} = 1$ if $(u, v) \in D$, $\mathcal{D}_{u,v} = 0$ otherwise, is constructed by running n breadth first searches, each rooted at a distinct node. This takes time $O(mn + n^2)$. Let G' be the graph obtained from G by replacing all directed arcs in G with an undirected arc. Matrix $\mathcal{U}$ such that $\mathcal{U}_{u,v} = 1$ if $(u, v) \in U$, 0 otherwise, can be constructed by running n breadth first searches, each rooted at a distinct node. This takes $O(mn + n^2)$ time.

There are $O(mn)$ 3-tuples of nodes for which inequalities (3.6),(3.10),(3.13) and (3.17) need to be generated. Similarly, there are $O(m^2n^2)$ 5-tuples of nodes for which inequalities (3.7), (3.11) need to be generated. There are $O(m^2n^3)$ 7-tuples for which (3.14), (3.15) and (3.16) need to be generated. There a re also $O(m)$ arcs such that the remaining inequalities need to be generated. Thus the MILP can be generated in (m^2n^3) time.

Often, due to cycle time considerations, the number of channels a process has is bounded by a constant k. In this case, the preceding analysis reveals that the MILP can be generated in $O(k^4n^3)$ time.

Pseudocode for generating this MILP is provided in Section 6.2

3.3.4 Multiple Scenarios

Recall that the MILP described thus far is constructed for a specific set of inputs to the circuit being slack matched. Each set of inputs is called a scenario. If there are multiple scenarios of concern to the designer, similar MILPs can be constructed for each scenario and solved simultaneously. Different sets of real variables are used in each of the MILPs. However, for each channel, the same integer variable is used to encode the number of slack matching buffers to be inserted on the channel. The set of MILP constraints for slack matching multiple scenarios is the union of the individual

sets of MILP constraints.

3.4 Results

The algorithm to generate a MILP equivalent to SMOP, from Section 3.3, was implemented in Modula-3 [21] and solved with `glpsol` [1], a freely available MILP solver. Two large examples were studied, the fetch loop of the Lutonium [19], an asynchronous 8051 micro-controller, and a control loop in the fetch unit of the MiniMIPS microprocessor [18].

3.4.1 Example I: Lutonium Fetch Loop

This algorithm was used to slack match the fetch loop of the Lutonium micro-controller. 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 8k instruction memory is modeled as $2s$ half buffers where s is the static slack of the memory.

Figure 3.8 shows the fetch loop of the Lutonium micro-controller. The 8051 instruction set contains instructions of different lengths(1,2,or 3 bytes). These instructions can be classified into four groups, those that read the instruction memory, those that write to the instruction memory, those that change the program counter (pc) and all other instructions. In the interests of cycle time, the Lutonium was designed so that two bytes of data are fetched from the instruction memory per cycle. I will describe the path of an instruction through this fetch loop. Initially, a pc is sent to the IMem, and two bytes are received from the IMem. A switch box(implemented by process SplD0,SplD1,MrgExt and MrgIO) routes the lower order byte to the IDecode process. This process determines the instruction type and length and forwards it to the state machine controller. The state machine controller consists of process IntCtrl, ExtControl,Router,CBUFE,CBUFIE,and CBUF. Based on the instruction type, the state machine controller sends a message to the switch box telling it how to route the higher order byte from the instruction memory. The collection of processes

PCUL,PCUH,PCIL,PCIH, PCNL and PCNH keep track of the current *pc*, and make any necessary changes to the current *pc*. The state machine controller also sends a message to these processes describing how the next *pc* is to be computed. For branch instructions, those that change the *pc*, the current *pc* needs to be forwarded to the Branch Unit. The new vale of the *pc* is also received from Branch Unit.

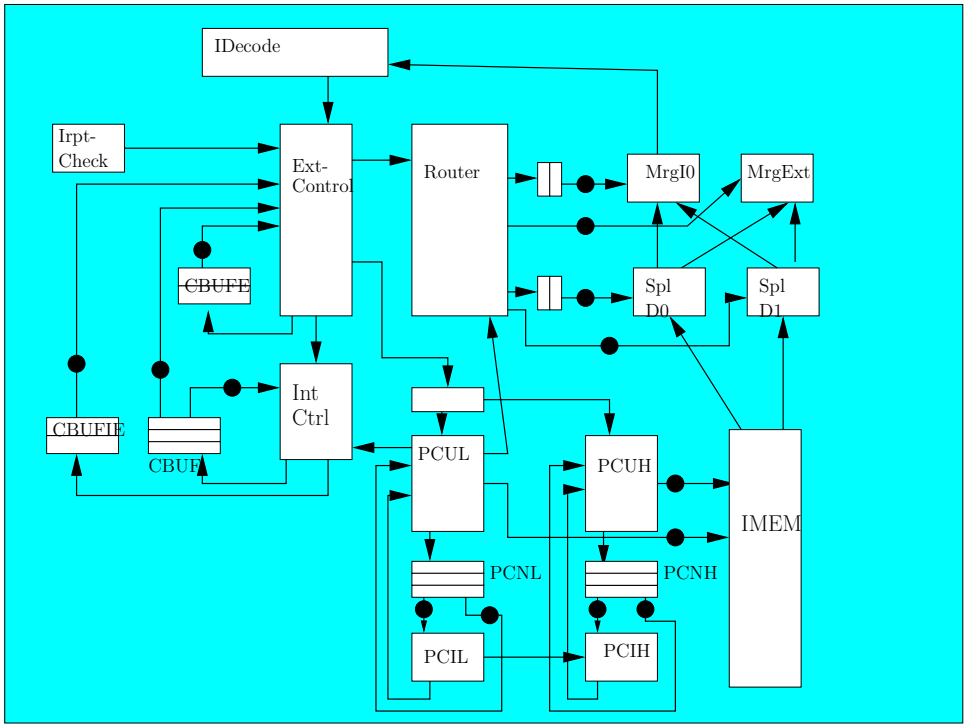


Figure 3.8: Lutonium fetch loop

Table 3.2 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. It took 0.1s to generate the MILP and 0.6s to solve the MILP for this circuit. The MILP was generated and solved on a machine with a 2.2 GHz Pentium 4 processor with 512 MB of memory. This MILP had 31 integer variables. The cycle time of the fetch was reduced from 68 transitions to 22 transitions.

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 3.2: Slack Matching buffers for Lutonium fetch

3.4.2 Example II: Control Loop of MiniMIPS

Figure 3.9 shows a loop in the fetch of the MiniMIPS.

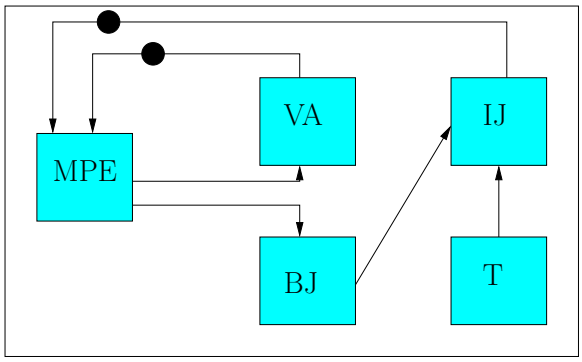


Figure 3.9: Control Loop in the MiniMIPS fetch

Table 3.3 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 both half buffers and full buffers was not included when generating the MILP. It took 0.1s to generate the MILP and 0.4s to solve it. The MILP was generated and solved on a machine with a 2.2 GHz Pentium 4 processor with 512 MB of memory. This MILP had 13 integer variables.

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

Table 3.3: Slack Matching buffers for the control loop in the MiniMIPS fetch

3.5 Conclusions and Future Work

A method of expressing SMOP as a MILP has been presented. This method 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 a MILP, and then solving using general purpose MILP solvers has been applied to circuits from the Lutonium [19] and the MiniMIPS [18]. Similar conditions to those in Sections 3.1 and 3.2 can be derived for slack matching circuits comprised of a restricted set of full buffers.

For the circuits studied so far, solving the MILP exactly has not taken a large amount of time. However, if solving the MILP for slack matching larger systems does take an excessive amount of time, heuristic MILP solvers should be considered [5]

Chapter 4

Slack Matching is NP-Complete

Whilst it has been conjectured that slack matching a QDI circuit is NP-complete [9, 22, 27], there is no proof in the literature that this problem is indeed NP-complete. Beerel [9] has shown that the pipeline optimization problem is NP-complete. Slack matching is a special case of the pipeline optimization problem. Beerel conjectures that slack matching is NP-complete. In this chapter, it is shown that slack matching is NP-complete via a reduction from the subset sum problem [7, SP13].

The slack matching decision problem is defined in Section 4.1. Next, in Section 4.2 a reduction from subset sum to slack matching is provided and it is shown that slack matching is NP-complete. The results are summarized in Section 4.3.

4.1 Slack Matching Decision Problem

NP is a class of decision problems. In this section, the decision problem corresponding to SMOP is defined. It will be shown that this decision problem is NP-complete.

Let S be the buffer that is to be added to channels in order to reduce the circuit's cycle time. S is called the slack matching buffer. If an SMOP has no feasible solution, then the circuit cannot be slack matched to cycle time τ_0. If the circuit has a feasible solution, the minimum value of $\sum_{\mathcal{X}} N(x_i)C(x_i)$ is said to be the cost of slack matching $(\mathcal{P}, \mathcal{X})$ to cycle time τ_0.

Definition 4.1 (Slack Matching Decision Problem(SMDP)). *Given $K \in [0, \infty)$ and a circuit $(\mathcal{P}, \mathcal{X})$, slack matching buffer S, target cycle time $\tau_0 \in \mathbb{R}^+$ and cost function*

C as in Definition **??**, *does there exist a mapping N as in Definition* **??** *such that* $\sum_{x_i \in \mathcal{X}} N(x_i)C(x_i) \leq K$.

4.2 NP Completeness of SMDP

In this section, it is show that SMDP is NP-complete by demonstrating a reduction from subset sum, a well known NP-complete problem, to SMDP. It is also shown that SMDP is in NP, completing the proof. The subset sum problem is defined as follows.

Definition 4.2 (Subset Sum)**.** *Given a set $\mathcal{K}$ of positive integers and positive integer T, does there exists a subset $\mathcal{S} \subseteq \mathcal{K}$ such that the sum of elements of $\mathcal{S}$ equals $\mathcal{T}$?*

4.2.1 Outline

This section is organized as follows. A variant of the subset sum problem is reduced to the problem of determining whether a particular circuit can be slack matched with cost at most H.

- In Section 4.2.2 the processes that are in the circuits used in the reduction are described.
- In Section 4.2.3, the circuit used in the reduction is described. This circuit is called a sum checker.

  A sum checker consists of ring comprised of a chain of circuits, called J-limiters, and two half buffers with an initial communication on the channel between the two half buffers. It is show in figure 4.1

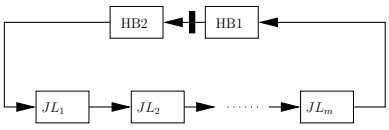


Figure 4.1: Sum Checker

  The reduction relies upon a circuit called a J-limiter. Such a circuit takes two parameters, H and J. The circuit has cycle time at most τ_0 if at most J slack

matching buffers have been inserted on a particular channel of the circuit and not slack matching buffers have been added on any other channel in the circuit. The cost of adding a slack matching buffer to any other channel in the J-limiter is $H + 1$.

- In Section 4.2.4, it is shown that slack matching is NP-hard. First it is proven that a variant of the subset sum problem is NP-complete. Next, it is shown how each instance of this problem can be reduced to a problem of slack matching a particular sum checker.

- Finally, in Section 4.2.5, it is shown that slack matching is NP-complete.

4.2.2 Class of Circuits Used in the Reduction

The circuit used in the reduction from subset sum to slack matching consists of processes that satisfy assumptions 3.1–3.5 (pp 37–38). Furthermore, it is assumed that the critical delay at τ_0 of paths between input and output transitions of the process are exactly the values in Table 3.1.

Let the paths between input and output variables of a process be classified as in Section 3.1.1. Theorem 3.1 shows that the critical delay of any cycle depends only on the number of $t^{i,j}(p)$ paths on the cycle for each process p with ports i and j. Thus it is sufficient to consider the collapsed constraint graph of a circuit comprised of such processes with only $t^{i,j}_{\uparrow\uparrow}(p)$ paths of each process p with ports i and j. In the following, when I refer to a collapsed constraint graph of a circuit, I mean the collapsed constraint graph of the circuit containing only edges that are on $t_{\uparrow\uparrow}$ paths of each process.

4.2.3 Sum Checkers

The subset sum problem is reduced to that of determining whether a circuit called a *sum checker* can be slack matched with cost at most H.

In Section 4.2.3.1, some properties of LR-buffers used in the reduction are stated.

The slack matching buffer used satisfies these properties. Next in Section 4.2.3.2, a circuit called a J-limiter is described. This circuit has the property that if it has cycle time at most τ_0 and the total cost of any slack matching buffers is at most J, at most one slack matching buffer has been added to a particular channel in the circuit, and no slack matching buffers have been added to any other channel in the circuit. In Section 4.2.3.3, the sum checker is described, and necessary and sufficient conditions for a sum checker to slack matched with cost at most H are stated.

4.2.3.1 Pipelines of LR-buffers

Let M be an LR-buffer such that the critical delays of paths, in its collapsed constraint graph, between input and output transitions are equal to the values in Table 3.1. Furthermore, let $f_M + b_M \leq \frac{\tau_0}{2}$ and $l_M + r_M \frac{\tau_0}{2}$. In Section 3.3.1, it was shown that in a pipeline of n such LR-buffers, the critical delay of any $\Phi_{\uparrow\uparrow}$ path is at most nf_M. Similarly, the critical delay of any $B_{\uparrow\uparrow}$ path is at most $nb_M - n\frac{\tau_0}{2}$, the critical delay of any $\Lambda_{\uparrow\uparrow}$ path is $l_M - \frac{\tau_0}{2}$ and the critical delay of any $P_{\uparrow\uparrow}$ path is r_M. It was also shown that there exist paths with exactly these critical delays.

4.2.3.2 J-limiters

In this section a circuit called a J-limiter is described. This circuit has the property that if it has cycle time τ_0 and the total cost of any slack matching buffers is at most J, then either no slack matching buffers have been introduced on any channel, or at most one buffer has been introduced a specific channel in the J-limiter, and no slack matching buffers have been introduced on any other channel. A J-limiter will be parametrized by three values J, H and τ_0 where J is as defined above, H is such that $H > J$, and $\tau_0 \geq 2J + 4$. Let S be a buffer with $f_S = r_S = 1$ and $l_S = b_S = \frac{\tau_0}{2} - y$ for any y such that $J + 1 \leq y$ and $y + 1 \leq \frac{\tau_0}{2}$. In the proceeding, we refer to S as the slack matching buffer.

Consider a circuit composed of processes as shown in Figure 4.2. Each node in the graph represents a process, and each edge a channel. Let the cost of introducing a slack matching buffer on all channels, except for (p_3, p_4), be $H + 1$. Let the cost of

introducing a slack matching buffer on channel (p_3, p_4) be J. Furthermore, there are no initial communications on any channel in this circuit.

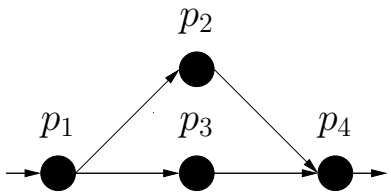


Figure 4.2: J-limiter

There is at most one channel between each pair of processes, thus each port of process p_k can be identified by the process at the other end of the channel to which the port belongs. For a process p_k, let the constants described in Table 3.1 be $f_k^{i,j}, b_k^{i,j}, l_k^{i,j}$ and $r_k^{i,j}$. For example, $f_2^{1,4}$ is critical delay of the $\phi_{\uparrow\uparrow}^{1,4}(p_2)$ path. Let $f_k^{i,j} = r_k^{i,j} = 1$ for all i,j and k. Similarly let $l_k^{i,j} = b_k^{i,j} = \frac{\tau_0}{2} - 1$ for all i and$j$, and $k \in \{1, 3, 4\}$. Furthermore, let $l_2^{1,1} = b_2^{4,1} = \frac{\tau_0}{2} - (J + 1)$ for $J > 1$ such that $J + 2 \leq \frac{\tau_0}{2}$.

The input port of p_1 is said to be the input port of the J-limiter. Similarly, the output port of p_4 is the output port of the J-limiter. For convenience, let the input port of the J-limiter be connected to process p_0 and the output port of the J-limiter be connected to process p_5.

Claim 4.1. *Consider a J-limiter such that n slack matching buffers have been added to channel (p_3, p_4), and no slack matching buffers have been added to any other channel. The circuit has cycle time τ_0 if and only if $n \leq J$.*

Proof. Consider the circuit obtained by adding n instances of S on channel (p_3, p_4). Its collapsed constraint graph, with only $t_{\uparrow\uparrow}^{i,j}(p)$ paths is shown in Figure 4.3. Edges marked by a broken line have critical delay zero. The collapsed constraint graph of each process is shaded, and labeled by process name. p_S is a pipeline of n instances of S inserted on channel (p_3, p_4). All nodes of the collapsed constraint graph are labeled by the name of the variable that the transition corresponds to. These names are local to each process. *li* and *lo* are respectively the input and output variables of an input port of a process. Similarly, *ri* and *ro* are the input and output variables

of an output port of a process. The subscript j is appended to a variable name if the process at the opposite end of the channel is process p_j. In order to distinguish variables in different processes with the same name, the prefix p_j is added to every variable of process p_j.

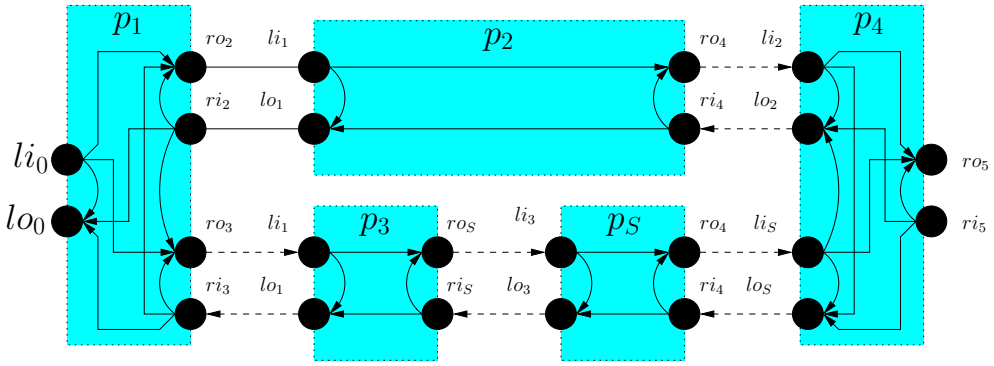


Figure 4.3: Constraint graph of J-limiter

The critical delay of each edge in the collapsed constraint graph is shown in Table 4.1.

Edge	Delay	Edge	Delay	Edge	Delay
$(p_1.li_0, p_1.ro_2)$	1	$(p_1.li_0, p_1.ro_3)$	1	$(p_1.li_0, p_1.lo_0)$	-1
$(p_1.ri_2, p_1.ro_2)$	1	$(p_1.ri_2, p_1.ro_3)$	1	$(p_1.ri_2, p_1.lo_0)$	-1
$(p_1.ri_3, p_1.ro_2)$	1	$(p_1.ri_3, p_1.ro_3)$	1	$(p_1.ri_3, p_1.lo_0)$	-1
$(p_2.li_1, p_2.ro_4)$	1	$(p_3.li_1, p_3.ro_S)$	1	$(p_S.li_3, p_S.ro_4)$	n
$(p_2.ri_4, p_2.ro_4)$	1	$(p_3.ri_S, p_3.ro_S)$	1	$(p_S.ri_4, p_S.ro_4)$	1
$(p_2.li_1, p_2.lo_1)$	$-J-1$	$(p_3.li_1, p_3.lo_1)$	-1	$(p_S.li_3, p_S.lo_3)$	-1
$(p_2.ri_4, p_2.lo_1)$	$-J-1$	$(p_3.ri_S, p_3.lo_1)$	-1	$(p_S.ri_4, p_S.lo_3)$	$-ny$
$(p_4.li_2, p_4.ro_5)$	1	$(p_4.li_S, p_4.ro_5)$	1	$(p_4.ri_5, p_4.ro_5)$	1
$(p_4.li_2, p_4.lo_2)$	-1	$(p_4.li_S, p_4.lo_2)$	-1	$(p_4.ri_5, p_4.lo_2)$	-1
$(p_4.li_2, p_4.lo_S)$	-1	$(p_4.li_S, p_4.lo_S)$	-1	$(p_4.ri_5, p_4.lo_S)$	-1

Table 4.1: Critical delay of edges in J-limiter's collapsed constraint graph

In order to prove the claim, the following must be shown.

1. If $0 \leq n \leq J$, the critical delay of all cycles in the collapsed constraint graph is at most zero.

2. If $n > J$, the critical delay of some cycle in the collapsed constraint graph is greater than zero.

Recall from Section 2.2 that there is no cycle with positive critical delay in a collapsed constraint graph, G, if and only if for each vertex u there exists x_u such that for any edge (u, v), $x_v \geq x_u + w(u, v)$ where $w(u, v)$ is the critical delay of edge (u, v). Note that each element of $\mathbf{x}$ corresponds to one vertex in the graph. For each transition, t, in the collapsed constraint graph of a J-limiter, it is can be verified that the following assignment to x_t satisfies this condition.

u	x_u	u	x_u	u	x_u	u	x_u
$p_1.li_0$	0	$p_1.lo_0$	-1	$p_1.ri_2$	0	$p_1.ro_2$	1
$p_1.ri_3$	0	$p_1.ro_3$	1	$p_2.li_1$	1	$p_2.lo_1$	0
$p_3.li_1$	1	$p_3.lo_1$	0	$p_3.ri_S$	1	$p_3.ro_S$	2
$p_S.li_3$	2	$p_S.lo_3$	1	$p_S.ri_S$	$1+n$	$p_3.ro_S$	$2+n$
$p_2.ri_4$	$1+n$	$p_2.ro_4$	$2+n$	$p_4.li_2$	$2+n$	$p_4.lo_2$	$1+n$
$p_4.li_S$	$2+n$	$p_4.lo_S$	$1+n$	$p_4.ro_5$	$3+n$	$p_4.ri_5$	$2+n$

Table 4.2: x_u such that for $0 \leq n \leq J$, $x_v - x_u \geq w(u, v)$

Note that this assignment does not satisfy the condition when $n > J$. When $n > J$ the cycle $\rho^{2,3}_{\uparrow\uparrow}(p_1), \phi^{1,S}_{\uparrow\uparrow}(p_3)\phi^{3,4}_{\uparrow\uparrow}(S)\lambda^{S,2}_{\uparrow\uparrow}(p_4)\beta^{4,1}_{\uparrow\uparrow}(p_2)$ has critical delay $n - J$. Since $n > J$, the cycle has positive critical delay. □

Consider a J-limiter that has been slack matched to τ_0, by inserting slack matching buffers only on channel (p_3, p_4). Next, the critical delays of all paths, in the collapsed constraint graph, between input and output transitions of such a J-limiter are determined.

Claim 4.2. *Consider a J-limiter with n slack matching buffers added on channel (p_3, p_4) such that the circuit's cycle time is at most τ_0. Recall that the slack matching buffer, S, is such that $f_S = r_S = 1$ and $b_S = l_S = \frac{\tau_0}{2} - y$ where $J + 1 \leq y$ and $y + 1 \leq \frac{\tau_0}{2}$. The critical delay of any $\Phi_{\uparrow\uparrow}, B_{\uparrow\uparrow}, \Lambda_{\uparrow\uparrow}$ or $P_{\uparrow\uparrow}$ path between input and output transitions of the J-limiter is at most:*

- $\Phi_{\uparrow\uparrow}$*:* $3 + n$,
- $B_{\uparrow\uparrow}$*:* $-(3 + \min\{ny, J\})$,
- $\Lambda_{\uparrow\uparrow}$*:* -1,

- $P_{\uparrow\uparrow}$: 1.

Furthermore, there exist paths with exactly these critical delays.

Proof. The claim is proven by considering all possible paths between input and output variables of the J-limiter. Only paths that traverse each edge in the collapsed constraint graph at most once need be considered.

I begin by proving that any $\Phi_{\uparrow\uparrow}$ path that traverses only ϕ edges satisfies the claim. There are two such paths:

1. $\phi^{0,2}_{\uparrow\uparrow}(p_1)\phi^{1,4}_{\uparrow\uparrow}(p_2)\phi^{2,5}_{\uparrow\uparrow}(p_4)$ and
2. $\phi^{0,3}_{\uparrow\uparrow}(p_1)\phi^{1,S}_{\uparrow\uparrow}(p_3)\phi^{3,4}_{\uparrow\uparrow}(p_S)\phi^{S,5}_{\uparrow\uparrow}(p_4)$.

The critical delays of these paths are respectively 3 and $3+n$. Thus, there exists a $\Phi_{\uparrow\uparrow}$ path with critical delay $3+n$. Any $\Phi_{\uparrow\uparrow}$ path must begin with a $\phi^{0,i}(p_1)$ path and end with a $\phi^{j,6}(p_3)$ path. If $i=j$, then the path is one of three paths listed above. Otherwise, the path must satisfy one of the following three conditions.

- The path contains a $\lambda^{i,j}_{\uparrow\uparrow}(p_4)$ path and no $\rho_{\uparrow\uparrow}(p_1)$ path.
- The path contains a $\rho^{i,j}_{\uparrow\uparrow}(p_1)$ path and no $\lambda_{\uparrow\uparrow}(p_4)$ path.
- The path contains at least one pair of paths $\lambda^{i,k}_{\uparrow\uparrow}(p_4)$ and $\rho^{m,j}(p_1)$ such that $i \neq k$ and $j \neq m$.

The critical delays of all $\lambda_{\uparrow\uparrow}(p_4)$ paths are equal. Similarly, the critical delays of all $\rho_{\uparrow\uparrow}(p_1)$ paths are equal, the critical delays of all $\phi_{\uparrow\uparrow}(p_1)$ paths are equal. Thus the critical delay of any such $\Phi_{\uparrow\uparrow}$ path is equal to the sum of the critical delay of a $\Phi_{\uparrow\uparrow}$ path containing only $\phi_{\uparrow\uparrow}$ paths and the critical delay of a cycle in the collapsed constraint graph of a J-limiter. Since the J-limiter is part of a circuit with cycle time τ_0, the critical delay of this cycle must be at most zero. This completes the proof for $\Phi_{\uparrow\uparrow}$ paths.

Similarly, there are two $B_{\uparrow\uparrow}$ paths that traverse only β edges. These are

1. $\beta^{5,2}_{\uparrow\uparrow}(p_4)\beta^{4,1}_{\uparrow\uparrow}(p_2)\beta^{2,0}_{\uparrow\uparrow}(p_1)$ and

2. $\beta^{5,S}_{\uparrow\uparrow}(p_4)\beta^{4,3}_{\uparrow\uparrow}(p_S)\beta^{S,1}_{\uparrow\uparrow}(p_3)\beta^{3,0}_{\uparrow\uparrow}(p_1)$.

Their critical delays are respectively $-3-J$ and $-3-ny$. respectively. Thus there exists a $B_{\uparrow\uparrow}$ path with critical delay at most $-(3+\min\{ny, J\})$.

Any $B_{\uparrow\uparrow}$ path must begin with a $\beta^{5,j}(p_4)$ path and end with a $\phi^{i,0}(p_1)$ path. If $i = j$, then the path is one of three past listed above. Otherwise, the path must satisfy one of the following three conditions.

- The path contains a $\lambda^{j,i}_{\uparrow\uparrow}(p_4)$ path and no $\rho_{\uparrow\uparrow}(p_1)$ path.
- The path contains a $\rho^{j,i}_{\uparrow\uparrow}(p_1)$ path and no $\lambda_{\uparrow\uparrow}(p_4)$ path.
- The path contains at least one pair of paths $\lambda^{j,k}_{\uparrow\uparrow}(p_4)$ and $\rho^{m,i}(p_1)$ such that $j \neq k$ and $i \neq m$.

The critical delays of all $\lambda_{\uparrow\uparrow}(p_4)$ paths are equal. Similarly, the critical delays of all $\rho_{\uparrow\uparrow}(p_1)$ paths are equal, the critical delays of all $\beta_{\uparrow\uparrow}(p_1)$ paths are equal. Thus the critical delay of any such $B_{\uparrow\uparrow}$ path is equal to the sum of the critical delay of a $B_{\uparrow\uparrow}$ path containing only $\beta_{\uparrow\uparrow}$ paths and the critical delay of a cycle in the collapsed constraint graph of a J-limiter. Since the J-limiter has cycle time τ_0, the critical delay of this cycle must be at most zero. This completes the proof for $B_{\uparrow\uparrow}$ paths.

There is only one $\Lambda_{\uparrow\uparrow}$ path traversing only $\lambda_{\uparrow\uparrow}$ edges. This path is $\lambda_{\uparrow\uparrow}(p_1)$ and has critical delay -1. All other $\Lambda_{\uparrow\uparrow}$ paths must begin with a $\phi^{0,i}_{\uparrow\uparrow}(p_1)$ path end with a $\beta^{j,0}_{\uparrow\uparrow}(p_1)$ path. Let L be such a path. Let C be the cycle in the J-limiter's collapsed constraint graph constructed by replacing the $\phi^{0,i}_{\uparrow\uparrow}(p_1)$ and $\beta^{j,0}_{\uparrow\uparrow}(p_1)$ edges in L by the $\rho^{j,i}_{\uparrow\uparrow}(p_1)$ edge. The critical delay of L must then be the sum of the the critical delay of $\phi^{0,i}_{\uparrow\uparrow}(p_1)$, the critical delay of $\beta^{j,0}_{\uparrow\uparrow}(p_1)$, and the difference between the critical delay of C and $\rho^{j,i}_{\uparrow\uparrow}(p_1)$. The sum of the critical delay of any $\phi_{\uparrow\uparrow}(p_1)$ and $\beta_{\uparrow\uparrow}(p_1)$ edge is zero. Since the J-limiter has cycle time τ_0, the critical delay of the cycle is at most zero. The critical delay of all $\rho_{\uparrow\uparrow}(p_1)$ edges is 1. Thus, the critical delay of all $\Lambda_{\uparrow\uparrow}$ paths must be at most -1.

A symmetric argument can be used to show that the critical delay of any $P_{\uparrow\uparrow}$ path is at most 1. There is only one $P_{\uparrow\uparrow}$ path traversing only $\rho_{\uparrow\uparrow}$ edges. This path is

$\rho_{\uparrow\uparrow}(p_3)$ and has critical delay 1. □

Next, the critical delays of paths between input and output variables of a pipeline of m J-limiters, when each J-limiter has cycle time τ_0, are derived. Each of the J-limiters may have different values J_i, but the values of H and τ_0 are the same across all J-limiters in the pipeline. Thus $H > J_i$ for all i. It is assumed that only one type of slack matching buffer, S, is used. Let y be such that $y \geq J_i$ for all i. Furthermore, let $\frac{\tau_0}{2} \geq y+1$ and $\frac{\tau_0}{2} > J_i+2$ for all i. As before, let $f_S = r_S = 1$ and $b_s = l_S = \frac{\tau_0}{2} - y$.

Claim 4.3. *Consider a pipeline of m J-limiters such that each J-limiter has cycle time τ_0 and all slack matching buffers have been added to only the (p_3, p_4) channel of each J-limiter. The critical delays of $\Phi_{\uparrow\uparrow}, B_{\uparrow\uparrow}, \Lambda_{\uparrow\uparrow}$ and $P_{\uparrow\uparrow}$ paths in this pipeline are*

- $\Phi_{\uparrow\uparrow}$*:* $3m + \sum_{i=1}^{m} n_i$,
- $B_{\uparrow\uparrow}$*:* $-\left(3m + \sum_{i=1}^{m} \min\{n_i y, J_i\}\right)$,
- $\Lambda_{\uparrow\uparrow}$*:* -1, *and*
- $P_{\uparrow\uparrow}$*:* 1,

where n_i denotes the number of slack matching buffers introduced on channel (p_3, p_4) of the i^{th} J-limiter, and J_i denotes the J parameter of the i^{th} J-limiter. Let the τ_0 and H parameters be identical across all J-limiters in the pipeline.

Proof. This claim is proven via induction. The base case of $m = 1$ holds by claim 4.2.

Assume that the claim is true for all pipelines of m_0 J-limiters. A $\Phi_{\uparrow\uparrow}$ path of any pipeline of $m_0 + 1$ J-limiters must be the composition a $\Phi_{\uparrow\uparrow}$ path of a pipeline of m_0 J-limiters and a $\Phi_{\uparrow\uparrow}$ path of a the $(m_0+1)^{th}$ J-limiter. The critical delay of this path is given by $3m_0 + \sum_{i=1}^{m_0} n_i + 3 + n_{m_0+1}$. This prove the claim for $\Phi_{\uparrow\uparrow}$ paths.

Similarly, a $B_{\uparrow\uparrow}$ path of any pipeline of m_0+1 J-limiters must be the composition a $B_{\uparrow\uparrow}$ path of a pipeline of m_0 J-limiters and a $B_{\uparrow\uparrow}$ path of a the $(m_0+1)^{th}$ J-limiter. The critical delay of this path is given by $-\left(3m_0 + \sum_{i=1}^{m_0} \min\{n_i y, J_i\}\right) - (3 + \min\{n_{m_0+1} y, J_{m_0+1}\})$. This proves the claim for $B_{\uparrow\uparrow}$ paths.

A $\Lambda_{\uparrow\uparrow}$ path of any pipeline of $m_0 + 1$ J-limiters is either a $\Lambda_{\uparrow\uparrow}$ path in the first J-limiter in the pipeline, or a composition of a $\Phi_{\uparrow\uparrow}$ path,$B_{\uparrow\uparrow}$ and $\Lambda_{\uparrow\uparrow}$ path such that the $\Phi_{\uparrow\uparrow}$ and $B_{\uparrow\uparrow}$ paths are in the first J-limiter and the $\Lambda_{\uparrow\uparrow}$ paths is in a pipeline of m_0 J-limiters. In the former case, the claim is proven by claim 4.2. In the latter case, by the inductive hypothesis, the critical delay of the $\Lambda_{\uparrow\uparrow}$ paths in a a pipeline of m_0 J-limiters is at most -1. Thus, the critical delay of the $\Lambda_{\uparrow\uparrow}$ path traversing $m_0 + 1$ J-limiters is at most $3 + n_1 - 1 - (3 + \min\{n_1 y, J_1\})$. Since $n_1 \leq J_1$ and $y \geq J_i$, the critical delay of this path is at most -1, which proves the claim for $\Lambda_{\uparrow\uparrow}$ paths.

Similarly, $P_{\uparrow\uparrow}$ path of any pipeline of $m_0 + 1$ J-limiters is either a $P_{\uparrow\uparrow}$ path in the last J-limiter in the pipeline, or a composition of a $\Phi_{\uparrow\uparrow}$ path,$B_{\uparrow\uparrow}$ and $P_{\uparrow\uparrow}$ path such that the $\Phi_{\uparrow\uparrow}$ and $B_{\uparrow\uparrow}$ paths are in the last J-limiter and the $P_{\uparrow\uparrow}$ path is in a pipeline of m_0 J-limiters. In the former case, the claim is proven by claim 4.2. In the latter case, by the inductive hypothesis, the critical delay of the $P_{\uparrow\uparrow}$ path in the pipeline of m_0 J-limiters is at most 1. Thus, the critical delay of the $P_{\uparrow\uparrow}$ path traversing $m_0 + 1$ J-limiters is at most $3 + n_{m_0+1} + 1 - (3 + \min\{n_{m_0+1} y, J_{m_0+1}\})$. Since $n_{m_0+1} \leq J_{m_0+1}$ and $y \geq J_{m_0+1}$, the critical delay of this path is at most 1, which proves the claim for $P_{\uparrow\uparrow}$ paths. □

Next, it is shown that if and only if each J-limiter in a pipeline of such J-limiters has cycle time τ_0 does the entire pipeline have cycle time τ_0.

Claim 4.4. *Consider a pipeline of m J-limiters such that slack matching buffers are only added to the (p_3, p_4) channel of each J-limiter. The pipeline has cycle time τ_0 if and only if each J-limiter in the pipeline has cycle time τ_0.*

Proof. If a particular J-limiter does not have cycle time τ_0, then its collapsed constraint graph must contain a cycle with positive critical delay. This cycle remains in the pipeline of J-limiters, thus the pipeline cannot have cycle time τ_0.

If all J-limiters in the pipeline have cycle time τ_0, then claim 4.3 holds. Any cycle in the constraint graph of the pipeline of m J-limiters can be expressed as the composition of a $\Lambda_{\uparrow\uparrow}$ path and a $P_{\uparrow\uparrow}$ path, each in pipelines of at most m J-limiters.

The critical delays of these paths must respectively be at most -1 and 1. Thus the critical delay of all cycles in the pipeline is non-positive. □

4.2.3.3 Sum Checkers

In this section, the circuit used in the reduction from subset sum to SMDP is described. Consider the circuit shown in Figure 4.4, where JL is a pipeline of m J-limiters, and TB is a pipeline of two LR-buffers, p_A, such that $f_A = r_A = 1$ and $b_A = l_A = \frac{\tau_0}{2} - 1$. Furthermore, let there be an initial communication on the channel between two instances of p_A in TB.

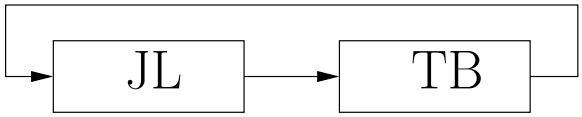


Figure 4.4: Circuit used in reduction from subset sum to SMDP

Consider the collapsed constraint graph of TB, with only $t^{i,j}_{\uparrow\uparrow}$ paths. There is only one simple $\Phi_{\uparrow\uparrow}$ path, and it has critical delay $2 - \tau_0$. Similarly, it can be shown that the critical $B_{\uparrow\uparrow}, \Lambda_{\uparrow\uparrow}$ and $P_{\uparrow\uparrow}$ paths of TB respectively have critical delay $-2 + \tau_0, -1$ and 1. Furthermore, there is only one cycle in this collapsed constraint graph and it has critical delay 0.

Let the slack matching buffer be S as described in Section 4.2.3.2. This can be achieved by choosing $y = \max_{i \in \{1 \ldots m\}} \{J_i\}$. Let the cost of adding a slack matching buffer to each channel within a J-limiter be as described in Section 4.2.3.2. Let the cost of adding slack matching buffers to channels between the J-limiters and channels with at last one port in TB be $H + 1$. In order for the all the process in the J-limiters to be implementable, the following inequalities must be satisfied for each process p_j:

- $\frac{\tau_0}{2} \geq f_j + b_j$
- $b_j > f_j$
- $f_j > 0$

In the construction $f_j = 1$ for all processes, thus the last constraint is satisfied for all processes. The slack matching buffer has $b_S = \frac{\tau_0}{2} - y$. Process p_2 in the i^{th} J-limiter has $b_2 = \frac{\tau_0}{2} - (J_i + 1)$. All other processes have $b = \frac{\tau_0}{2} - 1$. Since all $J_i > 1$, the first constraint in the list is satisfied for all processes. The second constraint imposes the following three inequalities:

- $\frac{\tau_0}{2} - y > 1$
- $\frac{\tau_0}{2} - (J_i + 1) > 1$
- $\frac{\tau_0}{2} - 1 > 1$

Recall $y = \max_{i \in \{1...m\}} \{J_i\}$. Thus, $\frac{\tau_0}{2} - y > 2$, or equivalently $\tau_0 > 2y + 4$ satisfies the above inequalities. Choosing $H > 2y$ and $\tau_0 = 3m + H + 2$ satisfies this constraint since $m \geq 1$.

Subset sum is reduced to slack matching such a sum checker.

Claim 4.5. *Consider a sum checker comprised of a pipeline of m J-limiters. Let the J parameter of the i^{th} sum checker be J_i. Let $y = \max_{i \in \{1...m\}} \{J_i\}$. Let all J-limiters have the same H parameter, H such that $H > 2y$. Consider a sum checker with cycle time $\tau_0 = 3m + H + 2$. Let the slack matching buffer, S be such that $f_S = r_S = 1$ and $l_S = b_S = \frac{\tau_0}{2} - y$. Let slack matching buffers be added only to (p_3, p_4) channels of each J-limiter in the sum checker, and to no other channels. The sum checker has cycle time τ_0 if and only if:*

$$2 + 3m + \sum_{i=1}^{m} n_i \leq 3m + H + 2 \leq 2 + 3m + \sum_{i:n_i \neq 0} J_i \tag{4.1}$$

$$0 \leq n_i \leq J_i \forall i \in \{1, \ldots, m\} \tag{4.2}$$

Proof. Assume towards contradiction that there exists a sum checker with cycle τ_0 and slack matching buffers added to only to the (p_3, p_4) channel of each J-limiter in the sum checker, and the above inequalities do not hold.

If (4.1) is not satisfied, consider the pair of cycles $\Phi_{\uparrow\uparrow}(JL)\Phi_{\uparrow\uparrow}(TB)$ and $B_{\uparrow\uparrow}(TB)B_{\uparrow\uparrow}(JL)$ in the collapsed constraint graph of the sum checker. From claim 4.3, the critical de-

lay of the former cycle is given by $3m + \sum_{i=1}^{m} n_i + 2 - \tau_0$. The critical delay of the latter cycle is given by $-3m - \sum_{i:n_i \neq 0} J_i - 2 + \tau_0$. Recall $\tau_0 = 3m + H + 2$. Since the circuit has cycle time τ_0, the critical delay of both cycles is at most zero. Thus

$$3m + \sum_{i=1}^{m} n_i + 2 \leq \tau_0 \leq 3m + \sum_{i:n_i \neq 0} J_i + 2$$

Substituting in for τ_0 shows that (4.1) holds.

If (4.2) is not satisfied for some i, then by claim 4.1, this J-limiter does not have cycle time τ_0, hence the sum checker cannot have cycle time τ_0.

Assume towards contradiction that there exist m, y, J_i, H and n_i such that the inequalities in the claim hold but the corresponding sum checker can not be slack matched by adding n_i slack matching buffers to the the the (p_3, p_4) channel of the i^{th} J-limiter in the sum checker.

By claim 4.1, each J-limiter must have cycle time τ_0. Thus by claim 4.4, the pipeline of J-limiters has cycle time τ_0. Similarly, TB has cycle time τ_0. Therefore the only cycles that could possibly have positive critical delay at τ_0 are cycles that traverse JL and TB. There are four such cycles:

- $B_{\uparrow\uparrow}(JL)B_{\uparrow\uparrow}(TB)$
- $\Phi_{\uparrow\uparrow}(JL)\Phi_{\uparrow\uparrow}(TB)$
- $\Lambda_{\uparrow\uparrow}(JL)P_{\uparrow\uparrow}(RB)$
- $P_{\uparrow\uparrow}(JL)\Lambda_{\uparrow\uparrow}(TB)$

From claim 4.3 and the analysis in Section 4.2.3.3, the critical delays of these cycles are respectively at most

- $-3m - \sum_{i:n_i \neq 0} J_i - 2 + \tau_0$
- $3m + \sum_{i=1}^{m} n_i + 2 - \tau_0$
- $-1 + 1$

- $1 + -1$

Furthermore, there exists cycles with exactly these critical delays. The last two cycles always have critical delay at most 0. Substituting $\tau_0 = 3m + H + 2$, if either $-3m - \sum_{i:n_i \neq 0} J_i - 2 + \tau_0 > 0$ or $3m + \sum_{i=1}^{m} n_i + 2 - \tau_0 > 0$, then either $3m + \sum_{i:n_i \neq 0} J_i + 2 < \tau_0 = 3m + H + 2$ or $\tau_0 = 3m + H + 2 < 3m + \sum_{i=1}^{m} n_i + 2$, reaching a contradiction. □

Lemma 4.1. *Consider a sum checker described in claim 4.5. Such a sum checker can be slack matched with cost at most H if and only if there exist $n_i \in \mathbb{N}$ such that the following inequalities hold:*

$$0 \leq n_i \leq J_i \quad \forall i \tag{4.3}$$

$$2 + 3m + \sum_{i=1}^{m} n_i \leq 3m + H + 2 \leq 2 + 3m + \sum_{i:n_i \neq 0} J_i \tag{4.4}$$

$$\sum n_i J_i \leq H \tag{4.5}$$

Proof. If there exist such n_i, then consider the circuit obtained by adding n_i slack matching buffers to the (p_3, p_4) channel of the i^{th} J-limiter in the sum checker. The cost of adding these slack matching buffers is given by $\sum n_i J_i$, which is at most H. From claim 4.5, the resulting circuit has cycle time τ_0.

If a sum checker can be slack matched with cost at most H, then no channel other than a (p_3, p_4) channel of a J-limiter has any slack matching buffers added to it, since the cost of adding a buffer to these channels is $H + 1$. Let n_i be the number of slack matching buffers added to the (p_3, p_4) channel of the i^{th} J-limiter. The cost of the slack matching buffers is given by $\sum n_i J_i$, which is assumed to be at most H. Since the sum checker has cycle time τ_0, by claim 4.5, (4.3) and (4.4) hold. □

4.2.4 SMDP is NP-Hard

In this section, it is shown that for any instance of the subset sum problem such that $\mathcal{T}$ is at least twice as large as the largest element in $\mathcal{K}$, a sum checker can be constructed such that it can be slack matched with cost at most H if and only if there

exists a subset of $\mathcal{K}$ such that the sum of its elements equals $\mathcal{T}$. Before describing this construction, it is proven that this restricted version of the subset sum problem is NP hard via a reduction from 3SAT.

In the following, $\mathcal{K}$ denotes the set of positive integers in an instance of subset sum, y the largest element of $\mathcal{K}$, and $\mathcal{S}$ and $\mathcal{T}$ are such that $\mathcal{S} \subseteq \mathcal{K}$ and the sum of elements of $\mathcal{S}$ equals $\mathcal{T}$. $\mathcal{T}$ is called the target of an instance of the subset sum problem.

Claim 4.6. *The subset sum problem, with the restriction that the target is at least $2y$, remains NP complete.*

Proof. This claim is proven via a reduction from 3SAT. Let C be the set of clauses of an instance of 3SAT, and U the set of variables. Let $m = |C|$ and $n = |U|$. In the following, all integers will be base 10, though any base greater than or equal to 4 suffices. Label the digits of the integers from 1 to $m + 2n$ in order from the least significant to most significant digit. For each variable $u_i \in U$, construct two $m + n$ digit integers as follows. The first integer corresponds to assigning the value **true** to u_i. Set digit i to 1 and for each clause c_j that contains the unnegated literal u_i, set digit $n + j$ to 1, and all remaining digits to 0. The second integer corresponds to the assigning **false** to u_i. Set digit i to 1 and for each clause c_j that contains the negated literal u_i, set digits $n + j$ to 1, and all remaining digits to 0. Additionally, for each clause c_j construct the following two integers. The first has a 1 in digit $n + j$ and 0 elsewhere. The second has a 2 in digit $n + j$ and 0 elsewhere. Let this set of integers be $\mathcal{K}$. Let the target integer contain a 1 for each digit $1 \ldots n$ and a 4 for each digit $n + 1 \ldots n + m$. The most significant digit of the target is 4. The largest element in $\mathcal{K}$ has a 2 as its most significant digit and 0 elsewhere. Let this integer be y. $2y$ has 4 as the most significant digit and 0 elsewhere. Thus, the target is strictly greater than $2y$.

Example 4.1. *Consider the 3SAT problem $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \neg x_4)$. The subset sum problem described above, that corresponds to this formula has target* 441111*. The set of integers is:*

	c_1	c_2	x_1	$x_2 4$	x_3	x_4
$x_1 =$ **true**	*1*	*0*	*0*	*0*	*0*	*1*
$x_1 =$ **false**	*0*	*0*	*0*	*0*	*0*	*1*
$x_2 =$ **true**	*0*	*1*	*0*	*0*	*1*	*0*
$x_2 =$ **false**	*1*	*0*	*0*	*0*	*1*	*0*
$x_3 =$ **true**	*1*	*1*	*0*	*1*	*0*	*0*
$x_3 =$ **false**	*0*	*0*	*0*	*1*	*0*	*0*
$x_4 =$ **true**	*0*	*0*	*1*	*0*	*0*	*0*
$x_4 =$ **false**	*0*	*1*	*1*	*0*	*0*	*0*
$(x_1 \vee \neg x_2 \vee x_3)$	*1*	*0*	*0*	*0*	*0*	*0*
$(x_1 \vee \neg x_2 \vee x_3)$	*2*	*0*	*0*	*0*	*0*	*0*
$(x_2 \vee x_3 \vee \neg x_4)$	*0*	*1*	*0*	*0*	*0*	*0*
$(x_2 \vee x_3 \vee \neg x_4)$	*0*	*2*	*0*	*0*	*0*	*0*

It remains to be shown that this subset sum problem has a solution if and only if the original 3SAT problem has a solution.

If the 3SAT instance is satisfiable, construct $\mathcal{S}$ as follows. Given a satisfying truth assignment, for each variable u_i select the integer that corresponds to u_i's value in the satisfying truth assignment and add it to the set $\mathcal{S}$. So far, $\mathcal{S}$ has exactly one integer with a non-zero entry for digit i for all $i \in \{1 \ldots n\}$. Furthermore, for each $j \in \{n+1 \ldots n+m\}$, $\mathcal{S}$ contains at least one integer with digit j equal to 1 and at most three such integers. The sum of the elements of $\mathcal{S}$ has value 1 in the n least significant digits. The remaining digits are between 1 and 3. If the digit is $n+j$ is 3, then add to $\mathcal{S}$ the integer with a 1 in only digit $n+j$, and 0 elsewhere. If digit $n+j$ is 2, then add the integer with a 2 in digit $n+j$ and 0 elsewhere. If digit $n+j$ is 1, then add both the integers with 0 in all digits but $n+j$. Thus, $\mathcal{S}$ has been constructed such that the sum of the elements of $\mathcal{S}$ is exactly the target.

Suppose there exists a subset of the integers that sums to the target. Note that there are at exactly two integers in $\mathcal{K}$ with a 1 at digit i such that $1 \leq i \leq n$. Similarly, there are exactly 4 integers with in $\mathcal{K}$with a 1 at digit j for j such that

$n < j \leq n + m$ and 1 integer with a 2 in this position. Since all integers are base 10, there are no carries when summing the members of any subset of $\mathcal{K}$. If a subset of integers sums to the target, then for each i, there is exactly one integer in $\mathcal{S}$ such that there is 1 in digit i, for $1 \leq i \leq n$. Consider the truth assignment obtained by setting u_i to **true** if this integer corresponds to the u_i having the value **true**, **false** otherwise. Note that for each $j : n+1 \leq j \leq m$, there must be at least one integer in $\mathcal{S}$ such that it contains at least two non-zero entries, one of which is at digit j. This integer corresponds to the truth assignment having a true literal in clause j. Thus, this assignment satisfies all clauses.

This proves that this problem is NP-hard. Since this problem is a restriction of subset sum, that is known to be in NP, this restricted version of subset-sum is NP complete. □

Lemma 4.2. *SMDP is NP-hard.*

Proof. The lemma is proven via a reduction from the restricted version of subset sum described in claim 4.6. Given an instance of subset sum such that $\mathcal{T} \geq 2y$ where y is largest element of $\mathcal{K}$, construct a sum checker as follows. Let the sum checker have $m = |\mathcal{K}|$ J-limiters. Let J_i be the J parameter of the i^{th} J-limiter. Set $J_i = k_i$ for each $k_i \in \mathcal{K}$. Choose $H = \mathcal{T}$. For the slack matching buffer, let $b_S = \frac{\tau_0}{2} - y$, where $\tau_0 = 3m + H + 2$. As shown in lemma 4.1, such a circuit can be slack matched with cost at most H if and only if there exist $n_i \in \mathbb{N}$ such that

$$0 \leq n_i \leq J_i \forall i$$

$$\sum n_i J_i \leq H$$

$$2 + 3m + \sum_{i=1}^{m} n_i \leq 3m + H + 2 \leq 2 + 3m + \sum_{i:n_i \neq 0} J_i$$

Rewriting the last inequality,

$$\sum_{i=1}^{m} n_i \leq H \leq \sum_{i:n_i \neq 0} J_i.$$

However,

$$\sum n_i J_i = \sum_{i:n_i \neq 0} n_i J_i \leq H \leq \sum_{i:n_i \neq 0} J_i.$$

Thus, $H = \sum_{i:n_i \neq 0} J_i$. Substituting for H and J_i,

$$\mathcal{T} = \sum_{i:n_i \neq 0} k_i.$$

Thus it has been shown that the sum checker described can be slack matched with cost at most τ_0 if and only if there exists $\mathcal{S} \subseteq \mathcal{K}$ such that the sum of the elements of $\mathcal{S}$ equals $\mathcal{T}$. If there is a solution to the subset sum problem, assigning n_i to 1 for each $k_i \in \mathcal{S}$ and 0 otherwise provides a solution to the slack matching problem. Given a solution to the slack matching problem, a solution to the subset sum problem can be constructed by choosing $\mathcal{S} = \{k_i : n_i \neq 0\}$. □

4.2.5 SMDP is NP-Complete

In this section, is shown that SMDP is in NP. This result, with that of lemma 4.2 proves that SMDP is NP-complete.

Lemma 4.3. *SMDP is in NP.*

Proof. It is easy to see that SMDP is a member of the class NP. The cycle time of a circuit can be determined in polynomial time by solving a linear program as demonstrated by Burns [3]. The collapsed constraint graph of a pipeline of n buffers can be expressed as a collapsed constraint graph with a fixed number(independent of n) of edges and vertices. The occurrence index offsets of the edges and the delays of the edges are a linear function of the number of buffers. Thus given the number of buffers n_i to place on each channel x_i, the cycle time of the circuit with n_i buffers on channel x_i can be determined in deterministic polynomial time. Similarly, the value of $\sum n_i s_i$ can be computed in linear time and compared to S. □

Theorem 4.1. *SMDP is NP-complete.*

Proof. The theorem follows directly from lemmas 4.2 and 4.3. □

4.3 Conclusions

It has been proven that slack matching is NP-complete via a reduction from subset sum. Subset sum is a problem that is NP-complete in the weak sense [7]. That is there exists a pseudo-polynomial time algorithm to solve the problem. A pseudo-polynomial time algorithm is one that is polynomial in the length of the encoding of its input, however there exists an exponentially smaller encoding of the input. The reduction presented shows that slack matching is NP-complete in the weak sense. It is not clear whether slack matching is also NP-complete in the strong sense.

Often times, the repetitive ER system of a circuit is constructed from a PRS in which the delay of all rules is identical. For such circuits, the size of the repetitive ER system is a linear function of the number of rules in the PRS. Furthermore the numeric value of all delays is at most 1, and the target cycle time has a numeric value smaller than the number of repeated rules in the repetitive ER system. Thus, if SMDP is NP-complete in the weak sense, a pseudo-polynomial time algorithm will be useful for slack matching circuits that are described as PRS.

Another related problem is that of determining whether a circuit can be slack matched, regardless of the cost of slack matching buffers. Determining whether this problem is NP complete remains an open problem.

Chapter 5

Slack Matching General QDI circuits

In Chapter 3, the SMOP for circuits composed entirely of a specified class of half buffers was expressed as a MILP. In this chapter, the SMOP for a larger class of circuits is expressed as a MILP. This MILP can be generated for any circuit that can be simulated by a repetitive ER system, provided the slack matching buffer used satisfies certain restrictions. It is assumed that all communication channels onto which slack matching buffers may be added implement a four phase handshake protocol, and that there are no initial communications on such channels. A MILP can be generated in a similar manner for other classes of LR-buffers used for slack matching.

It was shown in Section 2.2.1 that a repetitive ER system has cycle period τ_0 if and only if there exists $\mathbf{x}$ such that the system of inequalities (2.1) holds. This system of inequalities is:

$$\mathcal{A}\mathbf{x} \geq \mathbf{y}$$

where $\mathcal{A}$ is the arc-node incidence matrix of the collapsed constraint graph and each y_j is the critical delay at τ_0 of the j^{th} arc in the collapsed constraint graph. In this chapter, a MILP will be constructed that is equivalent to ensuring that the above inequality can be satisfied for the repetitive ER system of a circuit with $n_i \geq 0$ slack matching buffers added to channel i. The objective of the MILP is to minimize a weighted sum of n_i. This can be used, for example, to minimize the total energy consumed by the slack matching buffers.

The number of transitions and repeated rules in the repetitive ER system of a pipeline of n instances of a slack matching buffers is a linear function of n. The critical delay at τ_0 of each repeated rule is a constant. Thus the size of the system of inequalities (2.1) is a function of the variables to be solved for. It will be shown how the repetitive of ER system of a pipeline of $n \geq 0$ slack matching buffer can be represented by a repetitive ER system with a constant number of transitions. The critical delay of each repeated rule in this ER system is a linear function of the variables that encode n.

Recall that the repetitive ER system of a circuit is constructed from the repetitive ER systems of each process in the circuit as follows. For each pair of an output transition of a process, u, and the corresponding input transition of another process, v, add a repeated rule $\langle u, i-0\rangle \xrightarrow{0} \langle v, i\rangle$. A repetitive ER system of the circuit with n_i slack matching buffers added to each channel i is constructed by replacing the repeated rules described above, by a set of repeated rules that represent a pipeline of n_i slack matching buffers. The critical delay at τ_0 of such rules is a linear function of the variables of the encoding of n_i.

This chapter is organized as follows. First, in Section 5.1 the restrictions on the slack matching buffers are stated, and the critical delay at τ_0 between input and output transitions of $n \geq 1$ instances of S is determined. Next, in Section 5.2, a repetitive ER system with a constant number of transitions is constructed to represent a pipeline of $n \geq 0$ instances of S. The critical delay at τ_0 of a rule in this ER system is a linear function of the variables that encode n. Next, in Section 5.3, SMOP is formulated as a MILP. In Section 5.4, the results of solving an instance of the SMOP are described. This chapter is summarized in Section 5.5.

5.1 Pipelines of 1 or More LR Buffers

In this section the critical delays at τ_0 between input and output transitions of a pipeline of n instances of a LR buffer are determined provided the LR buffer satisfies the following assumption.

Assumption 5.1. *The LR buffer, S, used for slack matching has the following properties.*

- *S is a half buffer.*
- *S satisfies assumptions 3.4–3.5.*
- *There are no initial communications on any channel of S.*
- *$r_S + l_S \leq \frac{\tau_0}{2}$ and $f_S + b_S \leq \frac{\tau_0}{2}$ where l_S, r_S, b_S and f_S are specified as in Table 3.1. Since an LR buffer has only one input channel and one output channel, the superscripts of l_S, r_S, b_S and f_S are omitted.*

The last item in the list of property is required to ensure that the critical delay of all cycles in a pipeline of $n \geq 3$ instances of S have non-positive critical delay at τ_0. Consider the collapsed constraint graph of a pair of adjacent instances of S in a pipeline. There is a cycle in the collapsed constraint graph comprised of $\rho_{\uparrow\uparrow}$ path in one the instance of S and a $\lambda_{\uparrow\uparrow}$ path in the other instances of S. From Table 3.1, the critical delays of these paths are at most f_S and $l_S - \frac{\tau_0}{2}$. Example 3.1 shows an example of a buffer satisfying these assumptions.

In the remainder of this Chapter, let S be an LR-buffer satisfying assumption 5.1. Let the paths in the collapsed constraint graph of a pipeline of n instances of S be classified as in Section 3.1.1.

First, in Section 5.1.1, bounds on the critical delays of all paths between input and output transitions of a pipeline of $n \geq 1$ instances of S are derived. Next, in Section 5.1.2 these bounds are used to show that there is no cycle in the collapsed constraint graph of such a pipeline with positive critical delay at τ_0. The analysis in this Section is very similar to that in Section 3.1.

5.1.1 Critical Delays of Φ, B, Λ and P Paths in a Pipeline of Slack Matching Buffers

The following sequence of claims, bounds the critical delays of Φ, B, Λ and P paths between input and output transitions of a pipeline of $n \geq 1$ instances of S. Each

claim consists of two parts. The first part of the claim is a bound on the critical delay of a set of paths in a pipeline of $n \geq 1$ instances of S. The second part of the claim states when these bounds are tight. When the bounds are tight, the MILP for slack matching described in this chapter can be solved to determine the number of slack matching buffers to add in order to achieve the target cycle time whilst minimizing the cost function. When the bounds are not tight, the solution to the MILP determines a number of slack matching buffers to add to each channel in order to achieve the target cycle time. However, the cost function may not be minimized.

Claim 5.1. *The critical delay of any Λ path in a pipeline of $n \geq 1$ instances of S is at most:*

- $\Lambda_{\uparrow\uparrow}$*:* $l_S - \frac{\tau_0}{2}$
- $\Lambda_{\uparrow\downarrow}$*:* $l_S - \tau_0$
- $\Lambda_{\downarrow\uparrow}$*:* l_S
- $\Lambda_{\downarrow\downarrow}$*:* $l_S - \frac{\tau_0}{2}$

There exists a path with exactly these delays if the critical delays of paths between input and output transitions of S equal the values in Table 3.1.

Proof. Note that when the critical delays of the paths between input and output transitions of S equal the values in Table 3.1, the $\lambda_{\uparrow\uparrow}(S), \lambda_{\uparrow\downarrow}(S), \lambda_{\downarrow\uparrow}(S)$ and $\lambda_{\downarrow\downarrow}(S)$ paths have exactly the claimed critical delays.

The remainder of the claim is proven via induction. By assumption 3.4, the lemma holds for $n = 1$.

Assume that the claim holds for some $n = n_0$. Consider a pipeline of $n_0 + 1$ instances of S. If the Λ path traverses at most n_0 instances of the S, the claim is true by the inductive hypothesis.

Thus it remains to consider the case where the path traverses $n_0 + 1$ instances of S. In the following, each type of path $\Lambda_{\uparrow\uparrow}, \Lambda_{\uparrow\downarrow}, \Lambda_{\downarrow\uparrow}$ and $\Lambda_{\downarrow\downarrow}$ is considered separately.

Begin by considering a $\Lambda_{\uparrow\uparrow}$ path. Such a path falls into one of the following eight categories, based on the paths it traverses in the first buffer of the pipeline.

- The path traverses only a $\phi_{\uparrow\uparrow}$ and $\beta_{\uparrow\uparrow}$ path of the first buffer. The critical delay of the path is the sum of that of the $\phi_{\uparrow\uparrow}$ path, the $\beta_{\uparrow\uparrow}$ path and that of a $\Lambda_{\uparrow\uparrow}$ path in a pipeline of the remaining n_0 buffers. This is at most $f_S + b_S - \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2}$. By assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$, thus the critical delay of this path is at most

$$f_S + b_S - \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} \leq l_S - \frac{\tau_0}{2}.$$

- The path traverses only a $\phi_{\uparrow\downarrow}$ and $\beta_{\uparrow\uparrow}$ path of the first buffer. The critical delay of the path is the sum of that of the $\phi_{\uparrow\downarrow}$ path, the $\beta_{\uparrow\uparrow}$ path and that of a $\Lambda_{\downarrow\uparrow}$ path in a pipeline of the remaining n_0 buffers. This is at most $f_S - \frac{\tau_0}{2} + b_S - \frac{\tau_0}{2} + l_S$. By assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$, thus the critical delay of this path is at most

$$f_S - \frac{\tau_0}{2} + b_S - \frac{\tau_0}{2} + l_S \leq l_S - \frac{\tau_0}{2}.$$

- The path traverses only a $\phi_{\uparrow\uparrow}$ and $\beta_{\downarrow\uparrow}$ path of the first buffer. The critical delay of the path is the sum of that of the $\phi_{\uparrow\uparrow}$ path, the $\beta_{\downarrow\uparrow}$ path and that of a $\Lambda_{\uparrow\downarrow}$ path in a pipeline of the remaining n_0 buffers. This is at most $f_S + b_S + l_S - \tau_0$. By assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$, thus the critical delay of this path is at most

$$f_S + b_S + l_S - \tau_0 \leq l_S - \frac{\tau_0}{2}.$$

- The path traverses only a $\phi_{\uparrow\downarrow}$ and $\beta_{\downarrow\uparrow}$ path of the first buffer. The critical delay of the path is the sum of that of the $\phi_{\uparrow\downarrow}$ path, the $\beta_{\downarrow\uparrow}$ path and that of a $\Lambda_{\downarrow\downarrow}$ path in a pipeline of the remaining n_0 buffers. This is at most $f_S - \frac{\tau_0}{2} + b_S + l_S - \frac{\tau_0}{2}$. By assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$, thus the critical delay of this path is at most

$$f_S - \frac{\tau_0}{2} + b_S + l_S - \frac{\tau_0}{2} \leq l_S - \frac{\tau_0}{2}.$$

- The path traverses only a $\phi_{\uparrow\uparrow}$, a $\beta_{\uparrow\uparrow}$ and a $\rho_{\downarrow\downarrow}$ path of the first buffer. The critical delay of this path is the sum of that of the $\phi_{\uparrow\uparrow}$ path, the $\beta_{\uparrow\uparrow}$ path, the $\rho_{\downarrow\downarrow}$ path, a

$\Lambda_{\uparrow\downarrow}$ path in the pipeline of the remaining n_0 buffers, and a $\Lambda_{\downarrow\uparrow}$ path in a pipeline of the remaining n_0 buffers. This is at most $f_S + l_S - \tau_0 + r_S + l_S + b_S - \frac{\tau_0}{2}$ By assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$ and $r_S + l_S \leq \frac{\tau_0}{2}$, thus the critical delay of this path is at most

$$f_S + l_S - \tau_0 + r_S + l_S + b_S - \frac{\tau_0}{2} \leq l_S - \frac{\tau_0}{2}.$$

- The path traverses only a $\phi_{\uparrow\downarrow}$, a $\beta_{\uparrow\uparrow}$ and a $\rho_{\downarrow\uparrow}$ path of the first buffer. The critical delay of this path is the sum of that of the $\phi_{\uparrow\downarrow}$ path, the $\beta_{\uparrow\uparrow}$ path, the $\rho_{\downarrow\uparrow}$ path, a $\Lambda_{\downarrow\downarrow}$ path in the pipeline of the remaining n_0 buffers, and a $\Lambda_{\uparrow\uparrow}$ path in a pipeline of the remaining n_0 buffers. This is at most $f_S - \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + r_S + \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + b_S - \frac{\tau_0}{2}$ By assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$ and $r_S + l_S \leq \frac{\tau_0}{2}$, thus the critical delay of this path is at most

$$f_S - \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + r_S + \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + b_S - \frac{\tau_0}{2} \leq l_S - \frac{\tau_0}{2}.$$

- The path traverses only a $\phi_{\uparrow\uparrow}$, a $\beta_{\downarrow\uparrow}$ and a $\rho_{\uparrow\downarrow}$ path of the first buffer. The critical delay of this path is the sum of that of the $\phi_{\uparrow\uparrow}$ path, the $\beta_{\downarrow\uparrow}$ path, the $\rho_{\uparrow\downarrow}$ path, a $\Lambda_{\uparrow\uparrow}$ path in the pipeline of the remaining n_0 buffers, and a $\Lambda_{\downarrow\downarrow}$ path in a pipeline of the remaining n_0 buffers. This is at most $f_S + l_S - \frac{\tau_0}{2} + r_S - \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + b_S$ By assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$ and $r_S + l_S \leq \frac{\tau_0}{2}$, thus the critical delay of this path is at most

$$f_S + l_S - \frac{\tau_0}{2} + r_S - \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + b_S \leq l_S - \frac{\tau_0}{2}.$$

- The path traverses only a $\phi_{\uparrow\downarrow}$, a $\beta_{\downarrow\uparrow}$ and a $\rho_{\uparrow\uparrow}$ path of the first buffer. The critical delay of this paths is the sum of that of the $\phi_{\uparrow\downarrow}$ path, the $\beta_{\downarrow\uparrow}$ path, the $\rho_{\uparrow\uparrow}$ path, a $\Lambda_{\downarrow\uparrow}$ path in the pipeline of the remaining n_0 buffers, and a $\Lambda_{\uparrow\downarrow}$ path in a pipeline of the remaining n_0 buffers. This is at most $f_S - \frac{\tau_0}{2} + l_S + r_S + l_S - \tau_0 + b_S$ By assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$ and $r_S + l_S \leq \frac{\tau_0}{2}$, thus the critical delay of this

path is at most

$$f_S - \frac{\tau_0}{2} + l_S + r_S + l_S - \tau_0 + b_S \leq l_S - \frac{\tau_0}{2}.$$

This completes the proof for $\Lambda_{\uparrow\uparrow}$ paths. Symmetric arguments prove the claim for $\Lambda_{\uparrow\downarrow}, \Lambda_{\downarrow\uparrow}$ and $\Lambda_{\downarrow\downarrow}$ paths. □

Claim 5.2. *The critical delay of any P path in a pipeline of $n \geq 1$ instances of S is at most:*

- $P_{\uparrow\uparrow}$: r_S
- $P_{\uparrow\downarrow}$: $r_S - \frac{\tau_0}{2}$
- $P_{\downarrow\uparrow}$: $r_S + \frac{\tau_0}{2}$
- $P_{\downarrow\downarrow}$: r_S

There exists a path with exactly these delays if the critical delays of paths between input and output transitions of S equal the values in Table 3.1.

Proof. The proof of this claim is very similar to that of claim 5.1. Note that if the critical delays of paths between input and output transitions of S are exactly the values in Table 3.1, then the critical delays of $\rho_{\uparrow\uparrow}(S), \rho_{\uparrow\downarrow}(S), \rho_{\downarrow\uparrow}(S)$ and $\rho_{\downarrow\downarrow}(S)$ paths of the n^{th} buffer are exactly the specified values. Furthermore, any ρ path of the n^{th} instance of S is also a P path of the pipeline.

The remainder of the claim is proven via induction. By assumption 3.4, the lemma holds for $n = 1$.

Assume that the claim holds for some $n = n_0$. Consider a pipeline of $n = n_0 + 1$ instances of S. If the P path traverses at most n_0 instances of the S, the claim is true by the inductive hypothesis.

Thus it remains to consider the case where the path traverses $n_0 + 1$ instances of S. In the following, each type of path $P_{\uparrow\uparrow}, P_{\uparrow\downarrow}, P_{\downarrow\uparrow}$ and $P_{\downarrow\downarrow}$ is considered separately.

Begin by considering a $P_{\uparrow\uparrow}$ path. Such a path falls into one of the following eight categories, based on the paths it traverses in the n^{th} buffer of the pipeline.

- The path traverses only a $\beta_{\uparrow\uparrow}$ and a $\phi_{\uparrow\uparrow}$ path of the n^{th} buffer in the pipeline. In this case, the critical delay of the $P_{\uparrow\uparrow}$ path is the sum of that of the $\beta_{\uparrow\uparrow}$ path, a $P_{\uparrow\uparrow}$ path in a pipeline of the remaining n_0 buffers, and the $\phi_{\uparrow\uparrow}$ path. By the inductive hypothesis, this is at most $b_S - \frac{\tau_0}{2} + r_S + f_S$. By assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of such a $P_{\uparrow\uparrow}$ path is at most

$$b_S - \frac{\tau_0}{2} + r_S + f_S \leq r_S.$$

- The path traverses only a $\beta_{\uparrow\downarrow}$ and a $\phi_{\uparrow\uparrow}$ path of the n^{th} buffer in the pipeline. In this case, the critical delay of the $P_{\uparrow\uparrow}$ path is the sum of that of the $\beta_{\uparrow\downarrow}$ path, a $P_{\downarrow\uparrow}$ path in a pipeline of the remaining n_0 buffers, and the $\phi_{\uparrow\uparrow}$ path. By the inductive hypothesis, this is at most $b_S - \tau_0 + r_S + \frac{\tau_0}{2} + f_S$. By assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of such a $P_{\uparrow\uparrow}$ path is at most

$$b_S - \tau_0 + r_S + \frac{\tau_0}{2} + f_S \leq r_S.$$

- The path traverses only a $\beta_{\uparrow\uparrow}$ and a $\phi_{\downarrow\uparrow}$ path of the n^{th} buffer in the pipeline. In this case, the critical delay of the $P_{\uparrow\uparrow}$ path is the sum of that of the $\beta_{\uparrow\uparrow}$ path, a $P_{\uparrow\downarrow}$ path in a pipeline of the remaining n_0 buffers, and the $\phi_{\downarrow\uparrow}$ path. By the inductive hypothesis, this is at most $b_S - \frac{\tau_0}{2} + r_S - \frac{\tau_0}{2} + f_S + \frac{\tau_0}{2}$. By assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of such a $P_{\uparrow\uparrow}$ path is at most

$$b_S - \frac{\tau_0}{2} + r_S - \frac{\tau_0}{2} + f_S + \frac{\tau_0}{2} \leq r_S.$$

- The path traverses only a $\beta_{\uparrow\downarrow}$ and a $\phi_{\downarrow\uparrow}$ path of the n^{th} buffer in the pipeline. In this case, the critical delay of the $P_{\uparrow\uparrow}$ path is the sum of that of the $\beta_{\uparrow\uparrow}$ path, a $P_{\downarrow\downarrow}$ path in a pipeline of the remaining n_0 buffers, and the $\phi_{\uparrow\uparrow}$ path. By the inductive hypothesis, this is at most $b_S - \tau_0 + r_S + f_S + \frac{\tau_0}{2}$. By assumption 5.1,

$f_S + b_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of such a $P_{\uparrow\uparrow}$ path is at most

$$b_S - \tau_0 + r_S + f_S + \frac{\tau_0}{2} \leq r_S.$$

- The path traverses a $\beta_{\uparrow\uparrow}$, a $\phi_{\uparrow\uparrow}$ and a $\lambda_{\downarrow\downarrow}$ path of the n^{th} buffer in the pipeline. in this case, the critical delay of the $P_{\uparrow\uparrow}$ path is the sum of that of the $\beta_{\uparrow\uparrow}$ path, a $P_{\uparrow\downarrow}$ path in a pipeline of the remaining n_0 buffers, the $\lambda_{\downarrow\downarrow}$ path, a $P_{\downarrow\uparrow}$ path in a pipeline of the remaining n_0 buffers and the $\phi_{\uparrow\uparrow}$ path. By the inductive hypothesis, the critical delay of such a path is at most $b_S - \frac{\tau_0}{2} + r_S - \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + r_S + \frac{\tau_0}{2} + f_S$. However, by assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$ and $l_S + r_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of the $P_{\uparrow\uparrow}$ path is at most

$$b_S - \frac{\tau_0}{2} + r_S - \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + r_S + \frac{\tau_0}{2} + f_S \leq r_S.$$

- The path traverses a $\beta_{\uparrow\downarrow}$, a $\phi_{\uparrow\uparrow}$ and a $\lambda_{\downarrow\uparrow}$ path of the n^{th} buffer in the pipeline. in this case, the critical delay of the $P_{\uparrow\uparrow}$ path is the sum of that of the $\beta_{\uparrow\downarrow}$ path, a $P_{\downarrow\downarrow}$ path in a pipeline of the remaining n_0 buffers, the $\lambda_{\downarrow\uparrow}$ path, a $P_{\uparrow\uparrow}$ path in a pipeline of the remaining n_0 buffers and the $\phi_{\uparrow\uparrow}$ path. By the inductive hypothesis, the critical delay of such a path is at most $b_S - \tau_0 + r_S + l_S + r_S + f_S$. However, by assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$ and $l_S + r_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of the $P_{\uparrow\uparrow}$ path is at most

$$b_S - \tau_0 + r_S + l_S + r_S + f_S \leq r_S.$$

- The path traverses a $\beta_{\uparrow\uparrow}$, a $\phi_{\downarrow\uparrow}$ and a $\lambda_{\uparrow\downarrow}$ path of the n^{th} buffer in the pipeline. in this case, the critical delay of the $P_{\uparrow\uparrow}$ path is the sum of that of the $\beta_{\uparrow\uparrow}$ path, a $P_{\uparrow\uparrow}$ path in a pipeline of the remaining n_0 buffers, the $\lambda_{\uparrow\downarrow}$ path, a $P_{\downarrow\downarrow}$ path in a pipeline of the remaining n_0 buffers and the $\phi_{\downarrow\uparrow}$ path. By the inductive hypothesis, the critical delay of such a path is at most $b_S - \frac{\tau_0}{2} + r_S + l_S - \tau_0 + r_S + f_S + \frac{\tau_0}{2}$. However, by assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$ and $l_S + r_S \leq \frac{\tau_0}{2}$. Thus,

the critical delay of the $P_{\uparrow\uparrow}$ path is at most

$$b_S - \frac{\tau_0}{2} + r_S + l_S - \tau_0 + r_S + f_S + \frac{\tau_0}{2} \leq r_S.$$

- The path traverses a $\beta_{\uparrow\downarrow}$, a $\phi_{\downarrow\uparrow}$ and a $\lambda_{\uparrow\uparrow}$ path of the n^{th} buffer in the pipeline. in this case, the critical delay of the $P_{\uparrow\uparrow}$ path is the sum of that of the $\beta_{\uparrow\downarrow}$ path, a $P_{\downarrow\uparrow}$ path in a pipeline of the remaining n_0 buffers, the $\lambda_{\uparrow\uparrow}$ path, a $P_{\uparrow\downarrow}$ path in a pipeline of the remaining n_0 buffers and the $\phi_{\downarrow\uparrow}$ path. By the inductive hypothesis, the critical delay of such a path is at most $b_S - \tau_0 + r_S + \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + r_S - \frac{\tau_0}{2} + f_S + \frac{\tau_0}{2}$. However, by assumption 5.1, $f_S + b_S \leq \frac{\tau_0}{2}$ and $l_S + r_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of the $P_{\uparrow\uparrow}$ path is at most

$$b_S - \tau_0 + r_S + \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + r_S - \frac{\tau_0}{2} + f_S + \frac{\tau_0}{2} \leq r_S.$$

A symmetric argument can be used to prove the claim for $P_{\uparrow\downarrow}, P_{\downarrow\uparrow}$ and $P_{\downarrow\downarrow}$ paths.

□

Claim 5.3. *The critical delay of any Φ path in a pipeline of $n \geq 1$ instances of S is at most:*

- $\Phi_{\uparrow\uparrow}$*:* nf_S
- $\Phi_{\uparrow\downarrow}$*:* $nf_S - \frac{\tau_0}{2}$
- $\Phi_{\downarrow\uparrow}$*:* $nf_S + \frac{\tau_0}{2}$
- $\Phi_{\downarrow\downarrow}$*:* nf_S

There exists a path with exactly these delays if the critical delays of paths between input and output transitions of S equal the values in Table 3.1.

Proof. The proof of this claim is similar to that of claims 5.1 and 5.2.

Note that if the critical delays of paths between input and output transitions of S are exactly the values in Table 3.1, then by claim 3.1, there exist $\Phi_{\uparrow\uparrow}, \Phi_{\uparrow\downarrow}, \Phi_{\downarrow\uparrow}$ and $\Phi_{\downarrow\downarrow}$ paths in the pipeline with exactly the claimed delays.

The remainder of the claim is proven via induction.

Due to assumption 5.1, the claim is true for $n = 1$.

Assume that the claim hold for some $n = n_0$. It needs to be shown that the critical delay of any Φ path in a pipeline of $n = n_0 + 1$ instances of S satisfies the claim.

Consider the $\Phi_{\uparrow\uparrow}, \Phi_{\uparrow\downarrow}, \Phi_{\downarrow\uparrow}$ and $\Phi_{\downarrow\downarrow}$ paths separately. Begin by considering $\Phi_{\uparrow\uparrow}$ paths. Such a path must fall into one of the following six categories.

- The path traverses only a $\phi_{\uparrow\uparrow}$ path of the n^{th} process in the pipeline. In this case, the path is the composition of a $\Phi_{\uparrow\uparrow}$ path in a pipeline of the remaining n_0 instances of S with a $\phi_{\uparrow\uparrow}$ path in the n^{th} instance of S. By the inductive hypothesis and assumption 5.1, the critical delay of this path is at most $n_0 f_S + f_S = (n_0 + 1) f_S$.

- The path traverses only a $\phi_{\downarrow\uparrow}$ path of the n^{th} process in the pipeline. In this case, the path is the composition of a $\Phi_{\uparrow\downarrow}$ path in a pipeline of the remaining n_0 instances of S with a $\phi_{\downarrow\uparrow}$ path in the n^{th} instance of S. By the inductive hypothesis and assumption 5.1, the critical delay of this path is at most $n_0 f_S - \frac{\tau_0}{2} + f_S + \frac{\tau_0}{2} = n f_S$.

- The path traverses a $\lambda_{\downarrow\downarrow}$ and a $\phi_{\uparrow\uparrow}$ path of the n^{th} process in the pipeline. In this case, the path is the composition of a $\Phi_{\uparrow\downarrow}$ path in the first n_0 instances of S, followed by the $\lambda_{\downarrow\downarrow}$ path in the n^{th} instance of S, followed by a $P_{\downarrow\uparrow}$ path in the first n_0 instances of S, followed by a $\phi_{\uparrow\uparrow}$ path in the n^{th} instance of S. By the inductive hypothesis, claim 5.2 and assumption 5.1, the critical delay of such a path is at most $n_0 f_S - \frac{\tau_0}{2} + r_S + \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + f_S$. By assumption 5.1, $r_S + l_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of this path is at most

$$(n_0 + 1) f_S + r_S + l_S - \frac{\tau_0}{2} \leq (n_0 + 1) f_S.$$

- The path traverses a $\lambda_{\downarrow\uparrow}$ and a $\phi_{\uparrow\uparrow}$ path of the n^{th} process in the pipeline. In this case, the path is the composition of a $\Phi_{\uparrow\downarrow}$ path in the first n_0 instances of S, followed by the $\lambda_{\downarrow\uparrow}$ path in the n^{th} instance of S, followed by a $P_{\uparrow\uparrow}$ path

in the first n_0 instances of S, followed by a $\phi_{\uparrow\uparrow}$ path in the n^{th} instance of S. By the inductive hypothesis, claim 5.2 and assumption 5.1, the critical delay of such a path is at most $n_0 f_S - \frac{\tau_0}{2} + r_S + l_S + f_S$. By assumption 5.1, $r_S + l_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of this path is at most

$$(n_0 + 1)f_S + r_S + l_S - \frac{\tau_0}{2} \leq (n_0 + 1)f_S.$$

- The path traverses a $\lambda_{\uparrow\downarrow}$ and a $\phi_{\downarrow\uparrow}$ path of the n^{th} process in the pipeline. In this case, the path is the composition of a $\Phi_{\uparrow\uparrow}$ path in the first n_0 instances of S, followed by the $\lambda_{\uparrow\downarrow}$ path in the n^{th} instance of S, followed by a $P_{\downarrow\downarrow}$ path in the first n_0 instances of S, followed by a $\phi_{\downarrow\uparrow}$ path in the n^{th} instance of S. By the inductive hypothesis, claim 5.2 and assumption 5.1, the critical delay of such a path is at most $n_0 f_S + r_S + l_S - \tau_0 + f_S + \frac{\tau_0}{2}$. By assumption 5.1, $r_S + l_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of this path is at most

$$(n_0 + 1)f_S + r_S + l_S - \frac{\tau_0}{2} \leq (n_0 + 1)f_S.$$

- The path traverses a $\lambda_{\uparrow\uparrow}$ and a $\phi_{\downarrow\uparrow}$ path of the n^{th} process in the pipeline. In this case, the path is the composition of a $\Phi_{\uparrow\uparrow}$ path in the first n_0 instances of S, followed by the $\lambda_{\uparrow\uparrow}$ path in the n^{th} instance of S, followed by a $P_{\uparrow\downarrow}$ path in the first n_0 instances of S, followed by a $\phi_{\downarrow\uparrow}$ path in the n^{th} instance of S. By the inductive hypothesis, claim 5.2 and assumption 5.1, the critical delay of such a path is at most $n_0 f_S + r_S - \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + f_S + \frac{\tau_0}{2}$. By assumption 5.1, $r_S + l_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of this path is at most

$$(n_0 + 1)f_S + r_S + l_S - \frac{\tau_0}{2} \leq (n_0 + 1)f_S.$$

This proves the claim for $\Phi_{\uparrow\uparrow}$ paths. A symmetric argument can be used for $\Phi_{\uparrow\downarrow}, \Phi_{\downarrow\uparrow}$ and $\Phi_{\downarrow\downarrow}$ paths to complete the proof. □

Claim 5.4. *The critical delay of any B path in a pipeline of $n \geq 1$ instances of S is*

at most:

- $B_{\uparrow\uparrow}$: $nb_S - n\frac{\tau_0}{2}$
- $B_{\uparrow\downarrow}$: $nb_S - (n+1)\frac{\tau_0}{2}$
- $B_{\downarrow\uparrow}$: $nb_S - (n-1)\frac{\tau_0}{2}$
- $B_{\downarrow\downarrow}$: $nb_S - n\frac{\tau_0}{2}$

There exists a path with exactly these delays if the critical delays of paths between input and output transitions of S equal the values in Table 3.1.

Proof. The proof of this claim is very similar to that of claim 5.3.

Note that if the critical delays of paths between input and output transitions of S are exactly the values in Table 3.1, then by claim 3.2, there exist $B_{\uparrow\uparrow}, B_{\uparrow\downarrow}, B_{\downarrow\uparrow}$ and $B_{\downarrow\downarrow}$ paths in the pipeline with exactly the claimed delays.

The remainder of the claim is proven via induction.

Due to assumption 5.1, the claim is true for $n = 1$.

Assume that the claim holds for some $n = n_0$. It needs to be shown that the critical delay of any B path in a pipeline of $n = n_0 + 1$ instances of S satisfies the claim.

Consider the $B_{\uparrow\uparrow}, B_{\uparrow\downarrow}, B_{\downarrow\uparrow}$ and $B_{\downarrow\downarrow}$ paths separately. Begin by considering $B_{\uparrow\uparrow}$ paths. Such a path must fall into one of the following six categories.

- The path traverses only a $\beta_{\uparrow\uparrow}$ path of the first instance of S in the pipeline. In this case, the path is the composition of a $B_{\uparrow\uparrow}$ path in the last n_0 instances of S and a $\beta_{\uparrow\uparrow}$ path. By the inductive hypothesis and assumption 5.1, the critical delay of such a path is $b_S - \frac{\tau_0}{2} + n_0(b_S - \frac{\tau_0}{2}) = (n_0 + 1)(b_S - \frac{\tau_0}{2})$.

- The path traverses only a $\beta_{\downarrow\uparrow}$ paths of the first instance of S in the pipeline. In this case, the path is the composition of a $B_{\uparrow\downarrow}$ path in the last n_0 instances of S and a $\beta_{\downarrow\uparrow}$ path. By the inductive hypothesis and assumption 5.1, the critical delay of such a path is $b_S + n_0 b_S - (n_0 + 1)\frac{\tau_0}{2} = (n_0 + 1)(b_S - \frac{\tau_0}{2})$.

- The path traverses a $\rho_{\uparrow\uparrow}$ and $\beta_{\downarrow\uparrow}$ path of the first instance of S in the pipeline, and a $B_{\uparrow\uparrow}$ and a $\Lambda_{\uparrow\downarrow}$ path in he remaining n_0 instances of S. By the inductive hypothesis, assumption 5.1 and claim 5.1, the critical delay of such path is $n_0(b_S - \frac{\tau_0}{2}) + r_S + l_S - \tau_0 + b_S$. By assumption 5.1, $l_S + r_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of the path is

$$n_0(b_S - \frac{\tau_0}{2}) + r_S + l_S - \tau_0 + b_S \leq (n_0 + 1)(b_S - \frac{\tau_0}{2}).$$

- The path traverses a $\rho_{\uparrow\downarrow}$ and $\beta_{\downarrow\uparrow}$ path of the first instance of S in the pipeline, and a $B_{\uparrow\uparrow}$ and a $\Lambda_{\downarrow\downarrow}$ path in he remaining n_0 instance of S. By the inductive hypothesis, assumption 5.1 and claim 5.1, the critical delay of such path is $n_0(b_S - \frac{\tau_0}{2}) + r_S - \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + b_S$. By assumption 5.1, $l_S + r_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of the path is

$$n_0(b_S - \frac{\tau_0}{2}) + r_S - \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + b_S \leq (n_0 + 1)(b_S - \frac{\tau_0}{2}).$$

- The path traverses a $\rho_{\downarrow\uparrow}$ and $\beta_{\uparrow\uparrow}$ path of the first instance of S in the pipeline, and a $B_{\uparrow\downarrow}$ and a $\Lambda_{\uparrow\uparrow}$ path in he remaining n_0 instances of S. By the inductive hypothesis, assumption 5.1 and claim 5.1, the critical delay of such path is $n_0(b_S - \frac{\tau_0}{2}) - \frac{\tau_0}{2} + r_S + \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + b_S - \frac{\tau_0}{2}$. By assumption 5.1, $l_S + r_S \leq \frac{\tau_0}{2}$. Thus, the critical delay of the path is

$$n_0(b_S - \frac{\tau_0}{2}) - \frac{\tau_0}{2} + r_S + \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + b_S - \frac{\tau_0}{2} \leq (n_0 + 1)(b_S - \frac{\tau_0}{2}).$$

- The path traverses a $\rho_{\downarrow\downarrow}$ and $\beta_{\uparrow\uparrow}$ path of the first instance of S in the pipeline, and a $B_{\uparrow\downarrow}$ and a $\Lambda_{\downarrow\uparrow}$ path in he remaining n_0 instances of S. By the inductive hypothesis, assumption 5.1 and claim 5.1, the critical delay of such path is $n_0(b_S - \frac{\tau_0}{2}) - \frac{\tau_0}{2} + r_S + l_S + b_S - \frac{\tau_0}{2}$. By assumption 5.1, $l_S + r_S \leq \frac{\tau_0}{2}$. Thus, the

critical delay of the path is

$$n_0(b_S - \frac{\tau_0}{2}) - \frac{\tau_0}{2} + r_S + l_S + b_S - \frac{\tau_0}{2} \leq (n_0 + 1)(b_S - \frac{\tau_0}{2}).$$

□

5.1.2 Critical Delays of Cycles in a Pipeline of Slack Matching Buffers

Thus, the critical delays of paths between input and output transitions of a pipeline of $n \geq 1$ instances of S have been expressed as a linear function of the variables that encode n. Next is shown that any cycle in the collapsed constraint graph of a pipeline of $n \geq 1$ instances of S has non-positive critical delay at τ_0.

Claim 5.5. *The critical delay of any cycle in the collapsed constraint graph of a pipeline of* $n \geq 1$ *instances of* S *is at most* 0.

Proof. By assumption 5.1, the critical delay of any cycle entirely within the collapsed constraint graph of an instance of S is at most 0.

Note that there cannot be any cycles in the pipeline traversing only ϕ paths or only β paths. There must exist one instance of S in the pipeline such that this process traverses only ρ paths of this process. The cycle must fall into one of the following six cases.

- The cycle traverses only a $\rho_{\uparrow\uparrow}$ path of the process. The cycle must traverse a $\Lambda_{\uparrow\uparrow}$ path in the remainder of pipeline. In this case, the by claim 5.1 and assumption 5.1, the critical delay at τ_0 of the cycle is at most $l_S - \frac{\tau_0}{2} + r_S$. However, by assumption 5.1, $l_S + r_S \leq \frac{\tau_0}{2}$.

- The cycle traverses only a $\rho_{\uparrow\downarrow}$ path of the process. The cycle must traverse a $\Lambda_{\downarrow\uparrow}$ path in the remainder of pipeline. In this case, the by claim 5.1 and assumption 5.1, the critical delay at τ_0 of the cycle is at most $l_S + r_S - \frac{\tau_0}{2}$. However, by assumption 5.1, $l_S + r_S \leq \frac{\tau_0}{2}$.

- The cycle traverses only a $\rho_{\downarrow\uparrow}$ path of the process. The cycle must traverse a $\Lambda_{\uparrow\downarrow}$ path in the remainder of pipeline. In this case, the by claim 5.1 and assumption 5.1, the critical delay at τ_0 of the cycle is at most $l_S - \tau_0 + r_S + \frac{\tau_0}{2}$. However, by assumption 5.1, $l_S + r_S \leq \frac{\tau_0}{2}$.

- The cycle traverses only a $\rho_{\downarrow\downarrow}$ path of the process. The cycle must traverse a $\Lambda_{\downarrow\downarrow}$ path in the remainder of pipeline. In this case, the by claim 5.1 and assumption 5.1, the critical delay at τ_0 of the cycle is at most $l_S - \frac{\tau_0}{2} + r_S$. However, by assumption 5.1, $l_S + r_S \leq \frac{\tau_0}{2}$.

- The cycle traverses a $\rho_{\uparrow\uparrow}$ and $\rho_{\downarrow\downarrow}$ path of the process. The cycle must traverse a $\Lambda_{\uparrow\downarrow}$ and a $\Lambda_{\downarrow\uparrow}$ path in the remainder of the pipeline. By assumption 5.1 and claim 5.1, the critical delay of such a cycle is at most $r_S + l_S - \tau_0 + r_S + l_S$. However, by assumption 5.1, $l_S + r_S \leq \frac{\tau_0}{2}$.

- The cycle traverses a $\rho_{\uparrow\downarrow}$ and $\rho_{\downarrow\uparrow}$ path of the process. The cycle must Traver's a $\Lambda_{\downarrow\downarrow}$ and a $\Lambda_{\uparrow\uparrow}$ path in the remainder of the pipeline. By assumption 5.1 and claim 5.1, the critical delay of such a cycle is at most $r_S - \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2} + r_S + \frac{\tau_0}{2} + l_S - \frac{\tau_0}{2}$. However, by assumption 5.1, $l_S + r_S \leq \frac{\tau_0}{2}$.

In each case, the critical delay of the cycle is at most 0, proving the claim. □

5.2 Pipelines of 0 or More LR Buffers

In this section, it is shown how the collapsed constraint graph of a pipeline of $n : 0 \leq n \leq N$ instances of a slack matching buffer, S, can be represented by a collapsed constraint graph with size independent of n. Specifically, the number of nodes and arcs in the collapsed constraint graph is constant.

Let P and Q be two processes in a circuit that have a channel between them. Let $Q.ro$ and $Q.ri$ be the variables that implement the output port of Q of this channel. Similarly, let $P.li$ and $P.lo$ implement the input port of P of this channel. Let the channel be such that if a pipeline of $0 \leq n \leq N$ instances of S is inserted

Arc	Critical Delay	Arc	Critical Delay
$(Q.ro\uparrow, P.li'\uparrow)$	0	$(Q.ro\downarrow, P.li\downarrow)$	0
$(P.lo\uparrow, Q.ri\uparrow)$	0	$(P.lo\downarrow, Q.ri\downarrow)$	0

Table 5.1: Critical delays in the collapsed constraint graph of a pipeline of 0 instances of S.

Arc	Critical Delay	Arc	Critical Delay
$(Q.ro\uparrow, P.li\uparrow)$	nf_S	$(Q.ro\downarrow, P.li\downarrow)$	nf_S
$(Q.ro\uparrow, P.li\downarrow)$	$nf_S - \frac{\tau_0}{2}$	$(Q.ro\downarrow, P.li\uparrow)$	$nf_S + \frac{\tau_0}{2}$
$(P.lo\uparrow, Q.ri\uparrow)$	$n\left(b_s - \frac{\tau_0}{2}\right)$	$(P.lo\downarrow, Q.ri\downarrow)$	$n\left(b_s - \frac{\tau_0}{2}\right) - \frac{\tau_0}{2}$
$(P.lo\uparrow, Q.ri\downarrow)$	$n\left(b_s - \frac{\tau_0}{2}\right) + \frac{\tau_0}{2}$	$(P.lo\downarrow, Q.ri\uparrow)$	$n\left(b_s - \frac{\tau_0}{2}\right)$
$(P.lo\uparrow, P.li\uparrow)$	r_S	$(P.lo\downarrow, P.li\downarrow)$	r_S
$(P.lo\uparrow, P.li\downarrow)$	$r_S - \frac{\tau_0}{2}$	$(P.lo\downarrow, P.li\uparrow)$	$r_S + \frac{\tau_0}{2}$
$(Q.ro\uparrow, P.li\uparrow)$	$l_S - \frac{\tau_0}{2}$	$(Q.ro\downarrow, P.li\downarrow)$	$l_S - \tau_0$
$(Q.ro\uparrow, P.li\downarrow)$	l_S	$(Q.ro\downarrow, P.li\uparrow)$	$l_S - \frac{\tau_0}{2}$

Table 5.2: Critical delays in collapsed constraint graph of pipeline of $n \geq 1$ instances of S.

on this channel, $Q.ro$ and $Q.ri$ would be connected to the input port of this pipeline. Similarly, $P.li$ and $P.li$ would be connected to the output port of the pipeline. Such a pipeline of $0 \leq n \leq N$ instances of S can be represented by the following collapsed constraint graph. Let $E' = \{Q.ro\uparrow, Q.ri\uparrow, Q.ro\downarrow, Q.ri\downarrow, P.lo\uparrow, P.li\uparrow, P.lo\downarrow, P.li\downarrow\}$.

If $n = 0$, the set of nodes in the collapsed constraint graph of the pipeline is E'. The set of arcs is as in Table 5.1

If $n \geq 1$, claims 5.1– 5.4 specify the critical delays at τ_0 of all paths between input and output transitions of the pipeline. Claim 5.5 shows that any cycle internal to the collapsed constraint graph of this pipeline is at most 0. Thus, the transitions within this pipeline can be ignored. Therefore, a pipeline of $n \geq 1$ instances of S can be modeled by a collapsed constraint graph, whose set of nodes is E'. The set of arcs is as in Table 5.2.

Note that the set of arcs in this collapsed constraint graph is a superset of the set of arcs in that of a pipeline of $n = 0$ buffers. The critical delay at τ_0 of the arcs that appear in both collapsed constraint graphs is 0. The critical delay of each remaining arc is of the form $K_1 n + K_2$ where K_1 and K_2 are constants.

The collapsed constraint graph of a pipeline of $0 \leq n \leq N$ instances of S is represented by a collapsed constraint graph that has the same set of nodes and arcs as that representing a pipeline of $n \geq 1$ instances of S. However, the critical delays of these arcs is as described below.

If an arc appears in both the collapsed constraint graph of a pipeline of $n = 0$ buffers and that representing a pipeline of $n \geq 1$ buffers, the critical delay of the arc in the collapsed constraint graph representing a pipeline of $0 \leq n \leq N$ buffers is as in Table 5.2.

For any other arc in the collapsed constraint graph representing a pipeline of $n \geq 1$ instances of S, let the critical delay, from Table 5.2, be $K_1 n + K_2$. The critical delay of the corresponding arc in the collapsed constraint graph representing a pipeline of $0 \leq n \leq N$ buffers is chosen to be

$$K_1 n + K_2 z - K_3(1 - z) \tag{5.1}$$

where z is such that $z \leq n \leq zN$ and $z \in \{0, 1\}$. The value of K_3 depends upon the rest of the circuit that contains this pipeline of $0 \leq n \leq N$ instances of S. K_3 must be chosen in such a manner that if $n = 0$, then if a cycle traversing such an arc has positive critical delay, so does a cycle traversing only arcs that appear in Table 5.1.

Next, it is shown how the value of K_3 in (5.1) is chosen for each arc that is listen in Table 5.2 but not Table 5.1. For the arc $(Q.ro\uparrow, Q.ri\uparrow)$ K_3 can be chosen as the critical delay of the critical path in the collapsed constraint graph of P between $P.li\uparrow$ and $P.lo\uparrow$. When $n = z = 0$, if the critical delay of a cycle traversing the arc $(Q.ro\uparrow, Q.ri\uparrow)$ is positive, so is the critical delay of the cycle obtained by replacing the $(Q.ro\uparrow, Q.ri\uparrow)$ arc by the following sequence of arcs: a $(Q.ro\uparrow, P.li\uparrow)$ arc, the critical path between $P.li\uparrow$ and $P.lo\uparrow$, and a $(P.lo\uparrow, Q.ri\uparrow)$ arc. Similarly the value of K_3 for the arcs $(Q.ro\uparrow, Q.ri\downarrow), (Q.ro\downarrow, Q.ri\uparrow)$ and $(Q.ro\downarrow, Q.ri\downarrow)$ can be determined by respectively considering the critical delay of a path between:

- $P.li\uparrow$ and $P.lo\downarrow$,

- $P.li\downarrow$ and $P.lo\uparrow$, and
- $P.li\downarrow$ and $P.lo\downarrow$.

A symmetric argument can be used to show that the value of K_3 for the arcs $(P.lo\uparrow, P.li\uparrow), (P.lo\uparrow, P.li\downarrow), (P.lo\downarrow, P.li\uparrow)$ and $(P.lo\uparrow, P.li\uparrow)$ can be chosen respectively as the critical delay of the critical path between:

- $Q.ri\uparrow$ and $Q.ro\uparrow$,
- $Q.ri\uparrow$ and $Q.ro\downarrow$,
- $Q.ri\downarrow$ and $Q.ro\uparrow$, and
- $Q.ri\downarrow$ and $Q.ro\downarrow$.

It is next shown how the value of K_3 is chosen for the arc $(Q.ro\uparrow, P.li\downarrow)$. Let t be an output transition of P. Let B_t be the critical delay of the critical path between $P.li\downarrow$ and t. Similarly, let A_t be the critical delay of the critical path between $P.li\uparrow$ and t. Let v be the output transition of P such that the value of $B_v - A_v$ is maximized. K_3 is chosen to be $B_v - A_v$. If any cycle traverses the arc $(Q.ro\uparrow, P.li\downarrow)$, the cycle must contain a path between $P.li\downarrow$ and an output transition of P, say u. If such a cycle has positive critical delay, so does the cycle with the path between $P.li\downarrow$ and u replaced by the critical path between $P.li\downarrow$ and u. If $n = z = 0$, the cycle contains a path between $Q.ro\uparrow$ and u with critical delay $-K_3 + B_u$. There is a path between $Q.ro\uparrow$ and u that traverses the arc $(Q.ro\uparrow.P.li\uparrow)$ with critical delay A_u. K_3 was chosen such that $K_3 \geq B_u - A_u$ or equivalently, $A_u \geq -K_3 + B_u$. Thus, if such a cycle had positive critical delay, so does the cycle obtained by replacing the arc $(Q.ro\uparrow, P.li\downarrow)$ and the path between $P.li\downarrow$ and v by the arc $(Q.ro\uparrow, P.li\uparrow)$ and the critical path between $P.li\uparrow$ and v. The values of K_3 for the arcs $(Q.ro\downarrow, P.li\uparrow), (P.lo\uparrow, Q.ri\downarrow)$ and $(P.lo\downarrow, Q.ri\uparrow)$ can be chosen in a similar manner.

Note that if $K_3 \leq -K_2$ for all the arcs described above, there is no need to introduce the variable z and a pipeline of $n \geq 0$ instances of S can be represented by a collapsed constraint graph with nodes E' and arcs as in Table 5.2.

5.3 A MILP Formulation of Slack Matching

In this section, the SMOP for circuits composed of processes that can be simulated by repetitive ER systems is formulated as a MILP. It is assumed that all communications channels implement the four phase handshake protocol and that the slack matching buffer used satisfies assumption 5.1. Let S be such a slack matching buffer. Let $(\mathcal{P}, \mathcal{X})$ be the circuit that is to be slack matched. Let $(\mathcal{P}', \mathcal{X}')$ be the circuit obtained by inserting $0 \leq n_i \leq N$ slack matching buffers on each channel $x_i \in \mathcal{X}$. The SMOP is the problem of determining n_i such that a linear cost function of the n_i is minimized subject to the constraint that $(\mathcal{P}', \mathcal{X}')$ has cycle time at most τ_0. Since the collapsed constraint graph of a pipeline of $0 \leq n_i \leq N$ instances of S can be represented by a collapsed constraint graph with a constant size, as shown in Section 5.2, the system of constraints (2.1) can be constructed for the circuit $(\mathcal{P}', \mathcal{X}')$. Thus, the SMOP is equivalent to determining integers n_i such that this system of constraints can be satisfied whilst minimizing a linear cost function of n_i.

The construction of the system of constraint 2.1 is described below. In Chapter 1, the ports of a buffer process were classified as input ports or output ports. Let the ports of processes in a circuit to be slack matched be classified as follows. A port that can be connected to the input port of a slack matching buffer is an output port. Similarly a port that can be connected to an output port of a slack matching buffer is an input port.

Associate with each transition t of a process in $(\mathcal{P}, \mathcal{X})$ the real variable x_t. For each pair of transitions u and v in the same process, such that there is a repeated rule with source u and target v, there exists a constraint of the form

$$x_v - x_u \geq y_{uv} \tag{5.2}$$

where y_{uv} is the critical delay of the repeated rule with source u and target v. For each channel x_i, let u be an input variable of a port of the channel. Similarly, let v be an output variable of a port of the channel. For each such u and v, there exists a

constraint of the form

$$x_v - x_u \geq K_1^{uv} n_i + (K_2^{uv} + K_3^{uv}) z_i - K_3^{uv}$$

where K_1^{uv}, K_2^{uv} and K_3^{uv} are constants for each pair (u, v) as described in Section 5.2, $z_i \in \{0, 1\}$ and $z_i \leq n_i \leq N z_i$. This constraint can be rewritten,

$$x_v - x_u - K_1^{uv} n_i - (K_2^{uv} + K_3^{uv}) z_i \geq -K_3^{uv}. \tag{5.3}$$

If the variables z_i cannot be eliminated(as described at the end of Section 5.2, N can be chosen to be a large constant. Experience from the design of asynchronous micro-controllers [18, 19] has shown that the number of slack matching buffers required on any particular channel is small.

Observe that if the system of constraints (5.2)–(5.3) can be solved, then the critical delay at τ_0 of all cycles within the collapsed constraint graph of a process is at most 0. If the critical delay of all such cycles is indeed at most 0, the collapsed constraint graph of any process can be represented by a collapsed constraint graph containing only vertices corresponding to input and output transitions of the process. There is an arc between any pair of vertices in this collapsed constraint graph if and only if there is a path between the pair of vertices in the original collapsed constraint of the process. The critical delay of this arc is the critical delay of the critical path between the vertices in the original collapsed constraint graph. This transformation, when applied to all processes reduces the number of real variables x_u in the MILP formulation. Pseudocode for generating this MILP is described in Section 6.3

5.3.1 Multiple Scenarios

Recall that the repetitive ER system of a circuit can be constructed only for a specific set of inputs to the circuit to be slack matched. If the circuit has data, the same data values must be sent on each channel of the circuit. The MILP described in this chapter is based on a repetitive ER system. In order to allow for multiple scenarios, the same

approach as that in Section 3.3.4 can be used; this approach is as follows. If there are multiple scenarios of concern to the designer, similar MILPs can be constructed for each scenario and solved simultaneously. Disjoint sets of real variables are used for each of the MILPs. However, for each channel, the same integer variables are used to encode the number of slack matching buffers to be inserted on the channel. The set of MILP constraints for slack matching multiple scenarios is the union of the individual sets of MILP constraints.

5.4 Results

A tool was implemented to generate the MILP equivalent to SMOP formulated in this chapter. The tool was implemented in the C programming language. The MILP generated has been solved using `glpsol` [1]. This was applied to the fetch loop of the Lutonium [19], an asynchronous 8051 micro-controller. The fetch loop of the Lutonium was described in Section 3.4.1

The objective function minimized was the estimated energy consumption of the slack matching buffers.

This tool generates the MILP described in Section 5.3. This MILP does not require that each process in the circuit be a half buffer. Recall that in order to generate the MILP described in Section 3.3, the entire instruction memory needs to be modeled as a pipeline of half buffers. By contrast, in order to generate the MILP described in this chapter, the memory needs to be modeled as a repetitive ER system. All parts of the memory except the SRAM cells can be described as a PRS that can be modeled as a repetitive ER system. The SRAM cell used in the Lutonium contains pass transistors, which cannot be described by a PRS. Instead, the SRAM cell with pass transistors is modeled by an SRAM cell without pass transistors. This SRAM cell can be described as PRS that can be modeled as a repetitive ER system. A repetitive ER system of such an SRAM cell is generated, and the delays are chosen so as to model the delays of the SRAM cell with pass transistors.

The result of applying this slack matching algorithm is identical to that obtained

in Section 3.4.1. However, in this case only individual SRAM cells in the memory needed to be modeled, as opposed to modeling the entire memory(including logic surrounding the SRAM cells). It took 30s to generate the MILP and 2s to solve this MILP. The MILP was generated and solved on a machine with a 2.2 GHz Pentium 4 processor and 512 MB of memory. This MILP had 62 integer variables. The same reduction from 68 transitions to 22 transitions was obtained. Note that it took longer to generate the MILP than to solve it. The repetitive ER system of a circuit described as a collection of HSE can be constructed syntactically. However, in the absence of a HSE specification of a PRS, constructing an ER system from a PRS requires simulation. The index priority simulation described by Lee [10, Chapter 7] is an efficient method to construct the ER system of a PRS. Since the MILP was constructed from a PRS description of the circuit, it was necessary to perform index priority simulation to construct its repetitive ER system. If the HSE specification of a circuit is available, we expect it would take a substantially shorter amount of time to generate the MILP since there will be no need to perform an index priority simulation.

5.5 Summary

In this section, a method of formulating the SMOP as a MILP was presented. This formulation places no restrictions on the circuit to be slack matched except that it be possible to model each process as a repetitive ER system. Assuming the slack matching buffers used satisfy certain restrictions, it was shown how to formulate the SMOP as a MILP. This formulation was demonstrated on the fetch loop of the Lutonium micro-controller.

Chapter 6

Slack Matching in Practice

In the preceding chapters, two mixed integer linear programs equivalent to slack matching have been described. In this chapter, issues that arise when slack matching circuits are described. This chapter is divided into three sections. In the first section, it will be described how circuits composed entirely of half buffers can be slack matched by hand. In the remaining sections, it is described how the MILP described in Chapter 3 and Chapter 5 can be respectively be generated.

6.1 Slack Matching by Hand

Asynchronous QDI circuits are typically designed from an initial sequential description of the circuit. A set of program transformations is applied in order to produce an equivalent description of the circuit as a collection of concurrent processes that share information via message passing. A limited set of such transformations is applied repeatedly until each process can be implemented by a production rule set whose cycle time is below some target τ_0. The set of program transformations can often be applied in different orders, resulting in different circuits. The circuit designers' experience and intuition are used to determine the order in which the transformations are applied. Experience and intuition also determine when to stop applying these transformations. Oftentimes, the cycle time of the resulting circuit is much greater than that of process with the largest cycle time. Therefore, the resulting circuit needs to be slack matched. Typically, circuits are designed to minimize the energy consumed

while maintaining a cycle time of τ_0. Therefore, the energy the slack matching buffers consume needs to be computed in order to compare different designs. In this section, the MILP for slack matching circuits of half buffers derived in Chapter 3 is interpreted in terms of process graphs. In the remainder of this section, it will be assumed that all processes in a circuit to be slack matched satisfy the conditions described in Section 3.1.2. In Section 3.2, it was shown how to determine whether a circuit composed of such processes has cycle time at most τ_0 by examining undirected cycles in a circuit's process graph.

Recall that the process graph of a circuit is a graph with a node for each process. There is a directed arc in the process graph between nodes u and v if and only if there is a channel that consists of an output port of process u and input port of process v. For any arc (u, v) let $m_{u,v}$ be 1 if there is an initial communication on channel (u, v), 0 otherwise. For each pair of arcs $(u, v), (v, w)$ in the process graph let $\mathcal{F}_{uvvw} = \frac{f_v^{u,w}}{\tau_0} - m_{v,w}$ where $f_v^{u,w}$ is as in Table 3.1. Similarly let $\mathcal{B}_{wvvu} = \frac{1}{2} - \frac{b_v^{w,u}}{\tau_0} - m_{v,w}$ where $b_v^{w,u}$ is as in Table 3.1. For each ordered pair of arcs $(u, v), (w, v)$ let $\mathcal{L}_{uvvw} = \frac{1}{2} - \frac{l_v^{u,w}}{\tau_0}$ where $l_v^{u,w}$ is as in Table 3.1. Finally, for each ordered pair of arcs $(v, u), (v, w)$ let $\mathcal{R}_{uvvw} = \frac{r_v^{u,w}}{\tau_0} + m_{v,u} - m_{v,w}$ where $r_v^{u,w}$ is as in Table 3.1.

Let $G = (N, A)$ be a graph. Recall that an undirected cycle, c, in G is a sequence of ordered triples $\{(u_i, v_i, d_i)\}$ such that:

- $u_i, v_i \in N$,
- $d_i \in \mathbb{B}$,
- if $d_i = \textbf{true}$, $(u_i, v_i) \in A$,
- if $d_i = \textbf{false}$, $(v_i, u_i) \in A$, and
- for $i > 1$, $u_i = v_{i-1}$ and $u_1 = v_{\|c\|}$.

In Section 3.2, the following relation between an undirected cycle in a circuit's process graph and cycles in the circuit's collapsed constraint graph was shown.

Given an undirected cycle in the process graph, $c = \{(u_i, v_i, d_i)\}$, a corresponding cycle $c' = \{c'_i\}$ in the collapsed constraint graph is constructed as follows. For each pair of triplets $(u_i, v_i, d_i), (u_j, v_j, d_j) \in c$ such that either $j = i + 1$ or $i = \|c\| \wedge j = 1$:

- if $d_i = d_j = \mathbf{true}$, $c'_i \in \phi^{u_i,v_j}(v_i)$.
- if $d_i = d_j = \mathbf{false}$, $c'_i \in \beta^{u_i,v_j}(v_i)$.
- if $d_i = \mathbf{false} \wedge d_j = \mathbf{true}$, $c'_i \in \rho^{u_i,v_j}(v_i)$.
- if $d_i = \mathbf{true} \wedge d_j = \mathbf{false}$, $c'_i \in \lambda^{u_i,v_j}(v_i)$.

with the directions of the transitions chosen such that c' is indeed a cycle in the collapsed constraint graph.

Furthermore, theorem 3.1 provides a bound on the critical delay of any cycle in a circuit's collapsed constraint graph. This bound depends only on the number of $t^{u,w}(v)$ paths (for all t, u, v, w) in the cycle where $u, v, w \in N$ and $t \in \{\phi, \lambda, \rho, \beta\}$.

Let c be an undirected cycle in a process graph, (N, A). In the following, pairs of triples $(u_i, v_i, d_i), (u_j, v_j, d_j) \in c$ will be considered such that either $j = i + 1$ or $j = 1$ and $i = \|c\|$. Let c' be a directed cycle in the collapsed constraint graph of the circuit that corresponds to c. Furthermore let $w, x, y \in N$ For each c, the values $\mathcal{F}_{wxxy}, \mathcal{B}_{wxxy}, \mathcal{R}_{wxxy}$ and $\mathcal{L}_{wxxy}$ will be used to bound the critical delay of any corresponding c'. Let $FP(c)$ be the set of all ordered pairs of triples $(u_i, v_i, d_i), (u_j, v_j, d_j) \in c$ such that $d_j = d_i = \mathbf{true}$ such that either $j = i + 1$ or $j = 1$ and $i = \|c\|$. Let $BP(c)$ be the set of all ordered pairs of triples $(u_i, v_i, d_i), (u_j, v_j, d_j) \in c$ such that $d_j = d_i = \mathbf{false}$ such that either $j = i + 1$ or $j = 1$ and $i = \|c\|$. Let $LP(c)$ be the set of all ordered pairs of triples $(u_i, v_i, d_i), (u_j, v_j, d_j) \in c$ such that $d_j = \mathbf{false} \wedge d_i = \mathbf{true}$ such that either $j = i + 1$ or $j = 1$ and $i = \|c\|$. Let $RP(c)$ be the set of all ordered pairs of triples $(u_i, v_i, d_i), (u_j, v_j, d_j) \in c$ such that $d_j = \mathbf{true} \wedge d_i = \mathbf{false}$ such that either $j = i + 1$ or $j = 1$ and $i = \|c\|$.

$D(c)\tau_0$ is the bound on the critical delay of c' derived in theorem 3.1, where $D(c)$

is:

$$D(c) = \sum_{\substack{(u,v,d),(v,w,d') \\ \in FP(c)}} \mathcal{F}_{uvvw} - \sum_{\substack{(u,v,d),(v,w,d') \\ \in BP(c)}} \mathcal{B}_{uvvw} - \sum_{\substack{(u,v,d),(v,w,d') \\ \in LP(c)}} \mathcal{L}_{uvvw} + \sum_{\substack{(u,v,d),(v,w,d') \\ \in RP(c)}} \mathcal{R}_{uvvw}.$$

Thus, a circuit has cycle time at most τ_0 if for any undirected cycle in its process graph $D(c) \leq 0$.

6.1.1 Common Case of Slack Matching

Typically, when a circuit is designed by hand, a repetitive ER system of the circuit is constructed in the usual manner and the delay of each rule is set to one. Two asynchronous microprocessors [18, 19] have been designed using the PCHB template.

Let $G = (N, A)$ be the process graph of a circuit designed by hand using the PCHB template, that satisfies the restrictions in Section 3.1.2. In a typical implementation of the PCHB template $f_v^{u,w} = f$ for all $(u, v), (v, w) \in A$. Similarly, $b_v^{w,u} \geq f$ and $b_v^{w,u} + f \leq \frac{\tau_0}{2}$. For each pair of arcs $(u, v), (w, v) \in A$, $l_v^{u,w} \leq b_v^{x,w}$ for all x such that $(v, x) \in A$. Lastly, for each pair of arcs $(v, u), (v, w) \in A$, $r_v^{u,w} \leq f$. In the following, simplified conditions are presented that must be satisfied in order for such circuits to have cycle time at most τ_0. First, the undirected cycles in a process graph are classified into three groups.

- *Forward latency cycles*: A forward latency cycle is one such that $|BP(c)| = |LP(c)| = |RP(c)| = 0$. These cycles arise whenever there is a directed cycle in the process graph. Such cycles have $d_i =$ **true** for all $(u_i, v_i, d_i) \in c$, i.e. the undirected cycle traverses each edge in its direction. Figure 6.1 shows an example of such a cycle.

- *Backward latency cycles*: A backward latency cycle is one such that $|FP(c)| = |LP(c)| = |RP(c)| = 0$. These cycles arise whenever there is a directed cycle in the process graph. Such cycles have $d_i =$ **false** for all $(u_i, v_i, d_i) \in c$, i.e. the undirected cycle traverses each edge in the opposite direction. Figure 6.2 shows

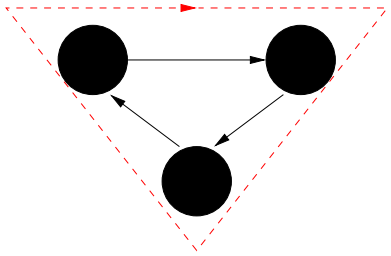


Figure 6.1: Forward latency cycle in a ring of buffers.

an example of such a cycle.

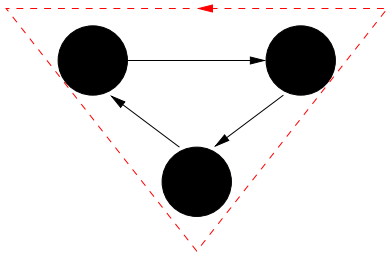


Figure 6.2: Backward latency cycle in a ring of buffers.

- *Cycles with direction change*: A cycle with direction change is one such that $|LP(c)| = |RP(c)| > 0$. The simplest case where such cycles arise is when there is more than one directed path from vertex u to v. Figure 6.3 shows an example of such a cycle.

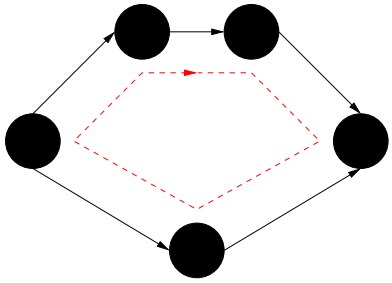


Figure 6.3: Cycle with direction change.

A circuit can be slack matched by inserting buffers in such a manner that for any undirected cycle, c, in the process graph of the resulting circuit $D(c) \leq 0$. Note that if there exists a forward latency cycle c' in the original circuit such that $D(c') > 0$,

then introducing slack matching buffers onto any channel along this cycle can only increase $D(c')$. Thus such circuits cannot be slack matched.

Having classified the undirected cycles in a collapsed constraint graph, the circuit structures that give rise to such cycles are described.

6.1.1.1 Rings of Processes

Consider the set of processes such traversed by a directed cycle in the process graph. Such a set of processes is called a ring of processes. The forward latency cycles and backward latency cycles impose constraints on the number of processes in such a ring.

Let c_f be a forward latency cycle. $D(c_f) = N\frac{f}{\tau_0} - m$ where N is the number of channels on the cycle and m is the number of channels with an initial communication. Therefore, if $D(c_f) \leq 0$,

$$Nf - m\tau_0 \leq 0 \iff \frac{Nf}{m} \leq \tau_0 \iff N \leq \frac{m\tau_0}{f}.$$

Recall from Chapter 1, that the number of channels of a ring with an initial communication is the number of initial messages on the ring. Thus, this inequality states that the cycle time is at least as large as rate at which messages move around the ring.

Let c_b be a backward latency cycle.

$$D(c_b) = -\sum_{\substack{(u,v,\textbf{false}),(v,w,\textbf{false}) \\ \in BP(c_b)}} \left(\frac{1}{2} - \frac{b_v^{u,w}}{\tau_0} - m_{w,v}\right).$$

Let the mean value of $b_v^{u,w}$ such that $(u, v, \textbf{false}), (v, w, \textbf{false}) \in BP(c_b)$ be $\bar{b}$. $D(c_b) = -N(\frac{1}{2} - \frac{\bar{b}}{\tau_0}) + m$ where N is the number of channels on the cycle and m is the number of channels with an initial communication. If $D(c_b) \leq 0$,

$$N\left(\frac{\tau_0}{2} - \bar{b}\right) - m\tau_0 \geq 0 \iff \tau_0 \geq \frac{N\bar{b}}{\frac{N}{2} - m} \iff N \geq \frac{2m\tau_0}{\tau_0 - 2\bar{b}}$$

Recall from Chapter 1 that since each process has finite static slack, the difference

between the rings' static slack and the number of messages it contains is the number of holes in the ring. This inequality states that the cycle time is at least as large as the rate at which holes move around the ring.

Thus, for any ring of processes, the forward latency cycle imposes an upper bound on the number of processes on the ring and the backward latency cycle imposes a lower bound on the number of processes. These bounds are:

$$\frac{2m\tau_0}{\tau_0 - 2\bar{b}} \leq N \leq \frac{m\tau_0}{f} \tag{6.1}$$

Example 6.1. *Consider a ring LR-buffer, each implemented using the PCHB template. Furthermore let there be one channel on this ring with an initial communication. Let the target cycle time, τ_0, be 22. Let $f = 2$. Typically, $3 \leq \bar{b} \leq 9$.*

If $\bar{b} = 9$, then,

$$\frac{2 \cdot 1 \cdot 22}{22 - (2 \cdot 9)} \leq N \leq \frac{1 \cdot 22}{2}$$
$$\iff 11 \leq N \leq 11.$$

Thus, there must be exactly 11 buffers on such a ring.

If $\bar{b} = 3$,

$$\frac{2 \cdot 1 \cdot 22}{22 - (2 \cdot 3)} \leq N \leq \frac{1 \cdot 22}{2}$$
$$\iff \frac{11}{4} \leq N \leq 11.$$

Thus, there must be at least 3 buffers on the ring and at most 11 buffers on the ring.

The value of $\bar{b}$ can be changed either be redesigning processes to reduce b, or adding buffers to the ring.

6.1.1.2 Fork and Join Paths

Let c_d be an undirected cycle in a process graph. Furthermore, let c_d be a cycle with direction change, that is $|LP(c_d)| = |RP(c_d)| \geq 1$. Substituting for $\mathcal{L}_{uvvw}$ and $\mathcal{R}_{uvvw}$

(from Section 6.1.1) and noting that $|LP(c)| = |RP(c)|$ for any undirected cycle c, it can be shown that for any undirected cycle c in a process graph:

$$-\sum_{\substack{(u,v,d),(v,w,d')\\ \in LP(c)}} \mathcal{L}_{uvvw} + \sum_{\substack{(u,v,d),(v,w,d')\\ \in RP(c)}} \mathcal{R}_{uvvw} \leq \sum_{\substack{(u,v,d),(v,w,d')\\ \in RP(c)}} m_{v,u} - m_{v,w}.$$

Thus, for an undirected cycle c in such circuits,

$$D(c) \leq \sum_{\substack{(u,v,d),(v,w,d')\\ \in FP(c)}} \mathcal{F}_{uvvw} - \sum_{\substack{(u,v,d),(v,w,d')\\ \in BP(c)}} \mathcal{B}_{uvvw} + \sum_{\substack{(u,v,d),(v,w,d')\\ \in RP(c)}} m_{v,u} - m_{v,w}.$$

Let c_d be an undirected cycle such that $|LP(c_d)| = |RP(c_d)|$. If

$$\sum_{\substack{(u,v,d),(v,w,d')\\ \in RP(c)}} m_{v,u} - m_{v,w} = 0,$$

then

$$D(c_d) \leq \sum_{\substack{(u,v,d),(v,w,d')\\ \in FP(c_d)}} \mathcal{F}_{uvvw} - \sum_{\substack{(u,v,d),(v,w,d')\\ \in BP(c_d)}} \mathcal{B}_{uvvw}.$$

The simplest scenario where an undirected cycle with direction change arises is when $|LP(c_d)| = |RP(c_d)| = 1$. An example of such a process graph is shown in Figure 6.3. Such a circuit structure is called a fork and join path, since there is a directed path in the process graph that splits into two forks which eventually join. Wong [27] calls this case reconvergent fanout.

Let $N_F = |FP(c_d)|$ and $N_B = |BP(c_d)|$. Furthermore, let

$$\sum_{\substack{(u,v,d),(v,w,d')\\ \in RP(c_d)}} m_{v,u} - m_{v,w} = 0, \tag{6.2}$$

and let

$$\sum_{\substack{(u,v,d),(v,w,d')\\ \in FP(c)}} m_{v,w} = \sum_{\substack{(u,v,d),(v,w,d')\\ \in FP(c)}} m_{v,u}. \tag{6.3}$$

Note that if there are no initial communications on any channel, the preceding conditions are satisfied. If these conditions hold,

$$D(c_d) \leq N_F \frac{f}{\tau_0} - \sum_{\substack{(u,v,d),(v,w,d') \\ \in BP(c_d)}} \left(\frac{1}{2} - \frac{b_v^{u,w}}{\tau_0} \right).$$

If $\bar{b}$ is the mean value of $b_v^{u,w}$ such that $(u,v,d),(v,w,d') \in BP(c_d)$, then

$$D(c_d) \leq N_F \frac{f}{\tau_0} - N_B \left(\frac{1}{2} - \frac{\bar{b}}{\tau_0} \right).$$

Thus, if N_F and N_B are such that

$$N_F \frac{f}{\tau_0} - N_B \left(\frac{1}{2} - \frac{\bar{b}}{\tau_0} \right) \leq 0 \iff \frac{N_F f + N_B \bar{b}}{\frac{N_B}{2}} \leq \tau_0, \tag{6.4}$$

then $D(c_d) \leq 0$.

Consider the process graph shown in Figure 6.4 and the highlighted undirected cycle. For the highlighted cycle, inequality (6.4) becomes

$$\frac{N_P f + N_Q \bar{b_Q}}{\frac{N_Q}{2}}. \tag{6.5}$$

For such a cycle, the sum of the number of holes and messages is determined by the initialization and remains constant. The holes reside in the pipeline Q and the messages in P. When process F inserts a message into P, it must also insert a message into Q. Thus, whenever a message is inserted into P, a hole is removed from Q. Similarly, whenever process J removes a message from P, it also removes one from Q. Thus, whenever a message is removed from P a hole is inserted into Q. The assumptions in Section 6.1.1 ensure that the delay between F removing a hole from Q and inserting a message in P and the delay between J removing a message from P and inserting a hole in Q are such that they need to be considered in the above analysis. When conditions (6.2) and (6.3) hold, inequality (6.5) states that the cycle time is at least as large as the average rate at which holes and tokens move around

the cycle.

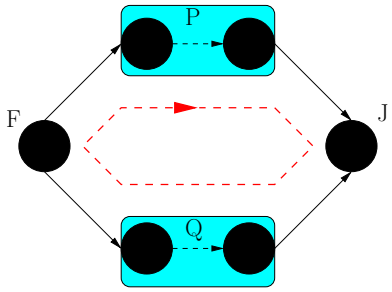


Figure 6.4: Example of a cycle with direction change.

Note that for any cycle c_d, a cycle c'_d can be constructed as follows. Let $c_d = \{(u_i, v_i, d_i)\}$. Let c'_d be the cycle such that $c'_d = \{(v_i, u_i, \neg d_i)\}$. That is, c'_d is the undirected cycle obtained by traversing each edge in the opposite direction to which it is traversed in c_d. Let $\bar{b'}$ be the mean value of $b_v^{u,w}$ such that $(u, v, d), (v, w, d') \in BP(c'_d)$. By construction of c'_d, $|BP(c'_d)| = |FP(c_d)|$ and $|FP(c'_d)| = |BP(c_d)|$. In a manner similar to that for c_d, it can be shown that $D(c'_d) \leq 0$ if

$$N_B \frac{f}{\tau_0} - N_F \left(\frac{1}{2} - \frac{\bar{b'}}{\tau_0} \right) \leq 0.$$

Consider such a pair of undirected cycles c_d and c'_d. Let N_F and N_B be defined as above. If conditions (6.2) and (6.3) hold, then $D(c_d) \leq 0$ and $D(c'_d) \leq 0$ if

$$N_B \left(\frac{2f}{\tau_0 - 2\bar{b'}} \right) \leq N_F \leq N_B \left(\frac{\tau_0 - 2\bar{b}}{2f} \right). \tag{6.6}$$

Recall that a source is a process with only output channels and a sink is a process with only input channels. Consider a circuit with only one source and one sink. If for each ring in the process graph condition (6.1) holds, and for each fork and join path in the process graph conditions (6.2), (6.3) and (6.6) hold, then the circuit has cycle time at most τ_0.

6.2 Slack Matching Circuits of Half Buffers

In this section, it is described how a circuit composed of half buffers can be slack matched using the method derived in Chapter 3. Pseudo-code for generating the described MILP is provided. The following input is required:

- process graph (N, A),
- the constants f, b, l and r as described in Table 3.1 for each process (specified by the mappings $F : A \times A \mapsto [0, \infty)$,$B : A \times A \mapsto [0, \infty)$,$L : A \times A \mapsto [0, \infty)$, and $R : A \times A \mapsto [0, \infty)$ respectively),
- a mapping $M : A \mapsto [0, 1]$ indicating the channels on which there are initial communications,
- a cost function $C : A \mapsto [0, \infty)$ specifying the cost of adding a slack matching buffer to a channel.
- $f_S^{i,j}$ and $b_S^{j,i}$ where S is the slack matching buffer, and
- τ_0, the target cycle time.

The process graph can be generated from the production rule set or HSE specifying the circuit. The mappings F, B, L and R can be determined from the ER system of each process in the circuit. Similarly, f_S and b_S can be determined from the ER system of the slack matching buffer.

Algorithm 1 shows how the matrices $\mathcal{U}$ and $\mathcal{D}$ can be generated.

Algorithm 2 shows how the MILP to be solved can be generated. The output of Algorithm 2 is the number of slack matching buffers to be added to each channel of the input circuit.

6.3 Slack Matching General QDI Circuits

In this section issues arising in the generation of the MILP described in Chapter 5 are discussed. Pseudo-code for generating this MILP is provided.

Algorithm 1 Generating the matrices $\mathcal{D}$ and $\mathcal{U}$

```
1: procedure GenU(N, A)
2:     A' ← A
3:     for each (u, v) ∈ A do
4:         INSERT(A', (v, u))
5:     end for
6:     return GEND(N,A')
7: end procedure
8: procedure GenD(N, A)    ▷ N-Nodes of process graph, A-arcs of process graph
9:     for  each u ∈ N do
10:        for  each v ∈ N do
11:            D[u, v] ← 0
12:        end for
13:    end for
14:    for each u ∈ N do
15:        color ← BFS(N, A, u)
16:        for each v ∈ N do
17:            if color[v]=black then
18:                D[u, v] ← 1
19:            end if
20:        end for
21:    end for
22:    return D
23: end procedure
24: procedure BFS(N,A,s)
25:    for each node u ∈ N − {s} do
26:        color[u]← white
27:    end for
28:    color[s]← gray
29:    Q ← s
30:    while Q ≠ do
31:        u ← head[Q]
32:        for each v ∈ Adj[u] do
33:            if color[v]=white then
34:                color[v]← gray
35:                ENQUEUE(Q,v)
36:            end if
37:        end for
38:        DEQUEUE(Q)
39:        color[u]← black
40:    end while
41:    return color
42: end procedure
```

Algorithm 2 Algorithm for generating MILP derived in Chapter 3

procedure GenMILP($N, A, M, C, f_S, b_S, \tau_0, F, B, L, R$)
    $D \leftarrow GEND(N, A)$
    $U \leftarrow GENU(N, A)$
    for each channel $(u, v) \in A$ **do**
        if $(v, u) \in D$ **then**
            Generate inequalities (3.8) and (3.12)
        end if
        Generate inequality (3.19)
        Generate inequalities (3.20)–(3.22)
    end for
    for each pair $(w, x), (x, y) \in A \times A$ **do**
        Generate inequalities (3.6) and (3.10)
        for each $(u, v) \in A : (v, w) \in D$ **do**
            Generate inequality (3.7)
        end for
        for each $(u, v) \in A : (y, u) \in D$ **do**
            Generate inequality (3.11)
        end for
    end for
    for each pair of channels $(w, x), (y, x) \in A \times A$ **do**
        Generate inequality (3.13)
    end for
    for each pair of channels $(x, w), (x, y) \in A \times A$ **do**
        Generate inequality (3.17)
        if $(w, y) \in U$ **then**
            Generate inequality (3.18)
        end if
    end for
    for all $u, v, w, x, y, z \in N : (w, x), (u, v), (z, y) \in A$ **do**
        if $\exists s \in N : (x, u), (v, s), (y, s) \in D$ **then**
            Generate inequality (3.15)
        end if
        if $\exists s \in N : (x, s), (v, s), (y, u) \in D$ **then**
            Generate inequality (3.14)
        end if
        if $\exists s \in N : (u, s) \in A \wedge (x, s), (v, x), (y, s) \in U$ **then**
            Generate inequality (3.16)
        end if
    end for
end procedure

The input of the algorithm is as follows.

- Closed production rule set P of the circuit to be slack matched, specified as a production rule set of each process and the connections between processes.
- A list, L of channels in the circuit. For each channel the following information is specified.
  - The name, n, of each channel variable in both ports of the channel.
  - A list, cv, of the channel variables specifying the variable in the slack matching buffer to which each channel variable can be connected.
  - The cost, c, of adding a slack matching buffer to the channel.
- The target cycle time, τ_0 of the circuit.
- The ER system, $\langle E_S, R_S \rangle$ of a pipeline of $n \geq 0$ slack matching buffers that has an constant number of rules, whose delays and occurrence index offsets are functions of n.

.

Algorithm 3 shows how the MILP described in Chapter 5 can be generated. This MILP can be solved to determine the variables n_l, the numbers of slack matching buffers that must be added to each channel of the circuit. The procedure IPS, is the index priority simulation algorithm described by Lee [10, Chapter 7] for generating an ER system from a production rule set.

Algorithm 3 Generating the MILP described in Chapter 5

1: $\langle E, R\rangle \leftarrow IPS(P)$
2: **for** each rule $\langle u, i-o\rangle \xrightarrow{\delta} \langle v, i\rangle \in R$ **do**
3: **if** $\nexists l \in L : u, v \in l.cv$ **then**
4: Generate inequality: $x_v - x_u \geq \delta - o \cdot \tau_0$
5: **end if**
6: **end for**
7: **for** each channel $l \in L$ **do**
8: Generate inequality $z_l \leq n_l \leq N$
9: $\triangleright$ N is a large constant
10: $\triangleright$ n_l is the number of slack matching buffers inserted on channel l
11: $\triangleright$ $z_l = 1$ is $n_l \geq 1$, $z_l = 0$ otherwise.
12: **for** each rule $\langle u, i-o\rangle \xrightarrow{\delta} \langle v, i\rangle \in R_S$ **do**
13: $\triangleright$ δ, o linear functions of n_l and z_l
14: u' is the transition in E' corresponding to u.
15: v' is the transition in E' corresponding to v.
16: Generate inequality $x_{v'} - x_{u'} \geq \delta - o \cdot \tau_0$
17: **end for**
18: **end for**

Chapter 7

Related Problems

In this Chapter several problems similar to slack matching QDI circuits are discussed. Each problem and its relation to slack matching is discussed.

In any synchronous distributed system, the cycle time of the system as a whole may be greater than that of each individual component due to the synchronization between various processes. However, not all such systems are slack elastic, that is, it is not always possible to add an arbitrary amount of buffering to each communication channel in the circuit without affecting the logical correctness of the system. However, for such systems, it may still be possible to add buffering to improve the cycle time provided the buffers are inserted in a manner that preserves the logical correctness.

This Chapter is organized as follows. First, in Section 7.1, it is described how slack matching information may be used as part of an automated synthesis flow for QDI circuits. Next, in Section 7.2, the problem of slack matching Bundled Data circuits is discussed. Finally, in Section 7.3, slack matching is compared to the retiming problem.

7.1 Clustering for $E\tau^2$

Wong [27] provides a method for synthesizing circuits such that each process can by implemented using the PCHB template. This synthesis method is called Data Driven Decomposition(DDD). Broadly speaking, DDD has two phases. The first phases consists of repeatedly applying a set of program transformations to a circuit

until they can no longer be applied. While this collection of processes can be directly implemented using the template, each process is usually much smaller than it needs to be. The second phase of DDD, called recomposition, groups these processes into clusters, such that each cluster describes a process that fits the target template. Wong [27, Chapter 5] describes a heuristic to explore the space of such clusterings and determine a clustering that consumes minimum energy subject to the constraint that the circuit's cycle time is at most τ_0.

QDI circuits do not use a global clock to synchronize the various processes. Instead, local handshakes are used. Let E be the energy consumed by a circuit and τ its cycle time. It has be shown that for QDI circuits, the energy consumed by the circuit can traded off for throughput via voltage scaling [20]. Furthermore, it has been observed that over a range of different voltages, the metric $E\tau^2$ is constant for a specific circuit. Oftentimes, a QDI designer is interested in minimizing $E\tau^2$ rather than minimizing E subject to a target cycle time τ_0. Prior to the recomposition phase, DDD does not use the target cycle time, τ_0.

Data Driven Decomposition(DDD) [27] is a method for the automated synthesis of QDI circuits from an initial sequential description. DDD consists of two phases. In the first phase, a finite set of program transformations is applied repeatedly to generate a set of concurrent processes that share information via message passing. This set of processes is slack elastic and cannot be decomposed further. In the second phase of DDD, these processes are grouped into clusters such that all the processes in a cluster can be collapsed into one process. Each cluster is directly implemented using the PCHB template. There are many such clusterings, thus this phase of DDD selects a clustering that minimizes some metric of the resulting circuit. Let E be the energy consumed by the resulting circuit and τ its cycle time. In this section, a method is described for determining clusters such that the $E\tau^2$ of the resulting circuit is small. Note that in addition to grouping processes into clusters, this phase of DDD also introduces LR-buffers for slack matching.

7.1.1 Circuit Templates

DDD produces a circuit that can be implemented using a target template. Wong [27] describes the DDD method for the PCHB template. Oftentimes, the energy consumed by a process can be estimated from the number of input channels and output channels a process has, and the encoding of the data values on each channel. This information can also be used to construct the collapsed constraint graph of the process.

A typical QDI process template is shown in Figure 7.1. The template has the

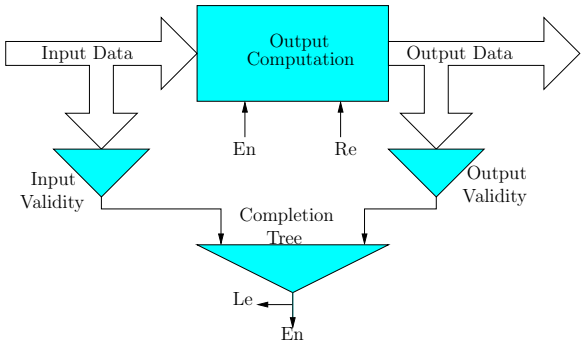


Figure 7.1: QDI process template

following three parts.

- *Output Computation*: This part of the circuit template computes values to send on the output port from the values received on input ports

- *Validity Checkers*: There is no global clock synchronizing the various processes. Therefore, when data is to be shared between two processes using boolean variables the data must be encoded in such a manner that it can determined whether the value of the variables is a valid codeword. Similarly there must be a neutral value that the variables are reset to between successive codewords. The transition from the neutral codeword to a valid codeword must be monotonic, i.e. when the variables change from the neutral value to a valid value, there are no intermediate states that are valid codewords. Each process must check the validity of the data received on input ports and output ports.

- *Completion Trees*: The validity(neutrality) of the individual channels are gath-

ered together by a completion tree. This component of the template is used in sequencing the communication actions on input and output channels.

7.1.2 Problem Definition

The first phase of DDD produces a network of processes that can be described by its process graph (N, A). Let M be a mapping $M : A \mapsto \{0, 1\}$ such that for any $(u, v) \in A$, $M(u, v) = 1$ if there is an initial communication on the channel (u, v), $M(u, v) = 0$ otherwise. Furthermore, let V be a mapping $V : A \mapsto [0, \infty)$ such that for any $(u, v) \in A$ $V(u, v)$ is the cost of computing the validity of a value on a port of the channel. Let S be a mapping $S : A \mapsto [0, \infty)$ such that for $(u, v) \in A$, $S(u, v)$ is the energy consumed by a slack matching buffer introduced on channel (u, v).

Let $G = (N, A)$ be a network of processes produced by the first phase of DDD, as described above. Let M, V and S be mappings as described above. Note that if G is produced by DDD, then there is at least on arc (u, v) on each directed cycle in G for which $M(u, v) = 1$. A clustering of G, M, V, N is a mapping $C : N \mapsto \mathbb{N}$ that satisfies the following restrictions. Let p be any directed simple path (one that traverse an arc at most once) in G from $v \in N$ to $w \in N$ such that $C(v) = C(w)$.

1. If all nodes, u, on the path p have $C(u) = C(v)$, there is not arc $(u', v') \in p$ such that $M(u', v') = 1$.

2. If p contains at least one arc (u', v') such that $C(u') \neq C(w)$. p contains at least one arc (u', v') such that $M(u', v') = 1$.

Let B be a mapping $B : A \mapsto \mathbb{N}$ such that for any $(u, v) \in A$ if $C(u) \neq C(v)$, $B(u, v)$ is the number of buffers introduced on arc (u, v) for the purposes of slack matching.

A cluster is a maximal subset of N such that all nodes u in the cluster have the same value $C(u)$. Consider a circuit obtained by mapping each cluster to a single process that fits the PCHB template. If the fanout of an output of a cluster is greater than 4, then additional copy processes are introduced for electrical reasons. For each

pair of u and v such that $(u, v) \in A$ and $C(u) \neq C(v)$ let the circuit contain $B(u, v)$ slack matching buffers on the channel (u, v) between the clusters $C(u)$ and $C(v)$. Let E be the energy consumed by the described circuit, and τ its cycle time. The second phase of DDD attempts to determine mappings C and B such that $E\tau^2$ is small.

7.1.3 Simulated Annealing

The optimization problem described in Section 7.1.2 is not convex. Thus, following Wong [27], the randomized heuristic of simulated annealing is used to determine an optimal clustering.

Simulated annealing is a randomized search heuristic. Associated with each point in the space to be searched is a cost function. The simulated annealing heuristic starts at an initial point in the space to be searched. Each iteration starts at a point in the search space. This starting point is perturbed. If the new point (the perturbed starting point) has lower cost than the initial point, the next iteration's starting point is this new point. If the cost of the new point is higher than the the starting point, with some non-zero probability the starting point of the next iteration is changed to the new point. Otherwise, the starting point of the next iteration is the starting point of the current iteration. The probability with which a point with greater cost is selected as the starting point of the next iteration varies with time. When this probability is changed, the starting point of the next iteration is set to the point traversed by the search that has the lowest cost. The manner in which this probability changes is called the annealing schedule.

In the remainder of this section, a simulated annealing heuristic is proposed to search the space of valid clusterings. First, the cost function is described. Next, it is described how the initial point in the search space is determined and its cost computed. Then the possible perturbations to a point in the search space are described. These perturbations move from one valid clustering to another. Finally a method for updating the costs is suggested.

Note that apart from computing the cost of the initial point in the search space,

simulated annealing only needs to compute the change in cost when moving from one point in the search space to the next. Thus this heuristic is particularly well suited to cost functions such that it is easy to compute the difference between the cost of two points in the search space.

7.1.3.1 Cost Function

A simulated annealing heuristic can be used to search the space of valid clusterings. The cost function considered is $E\tau^2$ where E is the energy consumed by the circuit and τ is its cycle time.

Since each process in the circuit either matches a template or is an instance of a slack matching buffer, the collapsed constraint graph of the circuit corresponding to a valid clustering can be determined. Let $\mathcal{A}$ be the arc-node incidence matrix of this graph. Let O and Δ be vectors that respectively specify the occurrence index offset and delay of each arc in the graph. Furthermore let $\mathcal{A}, O$ and Δ be ordered such that the occurrence index offset and delay of the arc in i^{th} row A are the values in the i^{th} elements of O and Δ respectively. The cycle time τ can be computed as the minimum τ that satisfies the following constraints

$$[\mathcal{A} \quad O]\,[\mathbf{x} \quad \tau]^T \geq \Delta.$$

Burns [3, Chapter 2] describes a primal-dual algorithm to solving this linear program that typically takes time linear in the number of nodes in the collapsed constraint graph.

E can be computed as the sum of the energy consumed by each process in the circuit. The energy consumed by each of the three components in a process that fits the template described in Section 7.1.1 can be estimated based upon the technology in which the circuit is to be implemented. The energy consumed by each process is simply the sum of the energies consumed by each of the individual components.

The energy consumed by the computation of the value to be sent on each output port is approximately equal. Thus, the energy used in computing all the output values

of a process is approximately linear in the number of output ports of a process.

For each type of code used on a communication channel, the energy consumed by a validity checker for the code can be measured. The total energy consumed by all the validity checkers in a process is then the sum of the energy consumed by the validity checker of each port of the process.

Typically, the energy consumed by the completion trees is linear in the number of ports a process has.

7.1.3.2 Initial Solution and Cost

A initial clustering can be constructed by choosing $B(u, v)$ to be 0 for all $(u, v) \in A$. $C(u)$ is chosen such that for any pair of distinct nodes, $u, v \in N$, $C(u) \neq C(v)$.

7.1.3.3 Moves

Next, the possible perturbations to clustering are described. The following set of moves is proposed. In the following let C and B be the mappings for starting point of an iteration of the simulated annealing heuristic. It is described how to construct C' and B' for a point that may be selected as the starting point of the next iteration.

- This move and the next correspond to moving a node at the boundary of a cluster to an adjacent cluster. Let $u, v \in N$ be such that $C(u) \neq C(v)$ and $(u, v) \in A$. Let $B' = B$ and $C'(u) = C(v)$ and for all $w \in N : u \neq w$ let $C'(w) = C(w)$ subject to the constraint that C' is a valid clustering.

- Let $u, v \in N$ be such that $C(u) \neq C(v)$ and $(u, v) \in A$. Let $B' = B$ and $C'(v) = C(u)$ and for all $w \in N : v \neq w$ let $C'(w) = C(w)$ subject to the constraint that C' is a valid clustering.

- This move and the next correspond to introducing an empty cluster and moving a node at the boundary of a cluster to the newly introduced cluster. Let $(u, v) \in A$ be such that $C(u) \neq C(v)$ and there exists $y \in N$ such that $C(y) = C(v)$ and $y \neq v$. For each $w \in N : w \neq v$ let $C'(w) = C(w)$. Select $C'(v) = a$ for some a such that there exists no $x \in N : x \neq v$ for which $C'(x) = a$.

- Let $(u, v) \in A$ be such that $C(u) \neq C(v)$ and there exists $y \in N$ such that $C(y) = C(u)$ and $y \neq u$. For each $w \in N : w \neq u$ let $C'(w) = C(w)$. Select $C'(u) = a$ for some a such that there exists no $x \in N : x \neq u$ for which $C'(x) = a$.

- The next two moves correspond respectively to increasing or decreasing the number of slack matching buffers on a channel between two clusters. Let $C' = C$. For one arc $(u, v) \in A$ such that $C(u) \neq C(v)$, let $B'(u, v) = B(u, v) + 1$. For all other arcs, $(u', v') \in A$ let $B'(u', v') = B(u', v')$.

- Let $C' = C$. For one arc $(u, v) \in A$ such that $C(u) \neq C(v)$, let $B'(u, v) = \min\{B(u, v) - 1, 0\}$. For all other arcs, $(u', v') \in A$ let $B'(u', v') = B(u', v')$.

- This move corresponds to merging two clusters. Let U and V be two distinct clusters. For any $u, v, w, x \in N : w,' \notin U \cup V, u, v \in U \cup V$ such that $(w, u), (v, x) \in A$, let p be a directed path in G from u to v that only traverses nodes in U or V. Let all such p contain at most one arc (y, z) such that $M(y, z) = 1$. For such a pair of clusters U and V and $u \in N$, let $B' = B$ and $C'(u) = C(u)$ if $u \notin U$. For $v \in V$ and $u \in U$ let $C'(u) = C(v)$.

- The last move corresponds to partitioning a cluster U into two cluster U_1 and U_2. Let $B' = B$ and U_1, U_2 be a partition of U. For each $u \in N : u \notin U_2$ let $C'(u) = C(u)$. Let $C'(u) = a$ for all $u \in U_2$ where a is such that there exists no $v \in N : C(v) = a$. Furthermore, U_1 and U_2 should be sch that C' is a valid clustering.

After one of the above moves is performed, the following modifications are made to the circuit in order to improve the electrical characteristics of the circuit and ensure that each cluster fits a template.

As described by Wong [27], the value sent on any pair of distinct arcs (u, v) and (u, w) is the same. For electrical reasons, if there are greater than four arcs (u, v_i) such that $C(u) \neq C(v_i)$ for any i and for all pairs i and j $C(v_i) \neq C(v_j)$ then a balanced tree of four way copy processes is inserted into the circuit on the channel

u of cluster $C(u)$ and the output of the tree of copy processes is sent to each cluster $C(v_j)$. For some circuits, an unbalanced copy tree may be preferable to a balanced copy tree due to slack matching reasons. Thus the insertion of unbalanced trees could be introduced as a second phase of moves in the simulated annealing.

If the template is a half buffer, it cannot contain an initial communication on both an input and output channel. Thus if a cluster has a pair of input and output channels such that there is an initial communication on each channel and the value of B of both channels is zero, then the value of B of the input channel is incremented by one.

The cost of a clustering can be evaluated by estimating the E and τ of the described circuit. This can be done by the method described in Section 7.1.3.1. Note that instead of evaluating the energy of each cluster, simply the energy of any cluster changed by a move needs to be computed.

Thus, a simulated annealing heuristic that can be used by DDD for the design of $E\tau^2$ optimal circuits has been proposed. A set of moves has been described and a method for computing the cost of such a clustering has been proposed. The circuit produced by the heuristic is slack matched.

7.2 Bundled Data Circuits

The slack matching problem as described arises in the design of bundled data asynchronous circuits. Such circuits consist of a control part and data part. The control part of the circuit is QDI. Delays are added to selected wires in the control part in order to ensure the logical correctness of the data path. Thus, the cycle time of such circuits is entirely determined by the cycle time of the control part of the circuit. If the control part is slack elastic, it may be slack matched in the manner described in Chapters 3 and 5.

7.3 Synchronous Circuits

Slack matching QDI circuits is often compared to retiming synchronous circuits. However, slack matching has been shown to NP-complete, whereas there exists a polynomial time algorithm for retiming [11].

All transitions in a synchronous VLSI circuit are synchronized to a global clock. At each 'tick' of the clock, each pipeline stage of the circuit latches in its inputs and produces an output at the subsequent clock tick. Such circuits are clearly not slack elastic. If a circuit has reconvergent fanout as shown in Figure 7.2, then the depth of pipelines P and Q must be changed by the same amount.

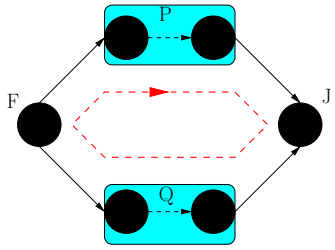


Figure 7.2: Example of Reconvergent Fanout

There is one lower bound on the cycle time of a synchronous circuit:

- The cycle time must be at least as large as the latency of an pipeline stage.

Retiming is an optimization to reduce the cycle time of a synchronous circuit. There is also the following lower bound on the cycle time of a synchronous circuit, that is intrinsic to the computation, and cannot be affected by retiming.

- The cycle time must be at least as large as the ratio of the delay around the cycle to the number of registers on this cycle.

Typically, a static timing analysis is performed to determine the cycle time of a synchronous circuit. Each static timing analysis can be performed in time polynomial in the size of the circuit. Whilst a retiming could potentially require an exponential number of static timing analyzes, Leiserson et al. [11] show retiming may be performed in polynomial time (in the size of the circuit). This is based on the observation that

retiming only changes the latency of adjacent pipeline stages. Thus, retiming can evaluate the cycle time of a synchronous circuit as the largest latency of any stage, that is the largest value of a set that is polynomial in the size of the circuit.

The cycle time of a QDI circuit is determined by considering all cycles in the collapsed constraint graph of the circuit. The cycle time is the largest ratio of delay along a cycle to the sum of occurrence index offsets on the cycle. As shown by Burns [3], the cycle time of a QDI circuit may be determined in time polynomial in the size of the circuit. Inserting slack matching buffers changes the delay of existing cycles in the circuit. It also introduces a set of new cycles, that did not exist in the original circuit. Unlike retiming, it has not been shown that the number of cycles introduced or changed by the insertion of a buffer is polynomial in the size of the circuit. Thus, slack matching must evaluate the cycle time as the maximum of an exponentially large set of values on each iteration.

In order for retiming not to affect the logical correctness of a synchronous circuit, the following constraints must be satisfied:

- The number of registers on any ring must remain constant.
- If the number of registers on a particular branch of reconvergent fanout is increased, the number of registers on all branches of this instance of reconvergent fanout must be increased by the same amount.

Thus, synchronous circuits are not slack elastic, i.e. an arbitrary number of registers cannot be added to the circuit without affecting its logical correctness. Since synchronous circuits are not slack elastic, there are fewer ways in which pipelining can be introduced, whilst preserving the logical correctness of the circuit. However, for a slack elastic QDI circuit, additional pipelining may be introduced on any channel whilst preserving the logical correctness of the circuit. Thus, the space of valid slack matchings is larger than the space of valid retimings for comparable circuits.

Chapter 8

Conclusion and Future Work

The slack matching problem has been formally described and two methods of slack matching QDI circuits have been presented. Both of these methods rely on constructing a mixed integer linear program(MILP) that is equivalent to slack matching this circuit. It has been shown that slack matching is in general an NP-complete problem, via a reduction from subset sum. Finally the impact of slack matching on the design of QDI circuits has been discussed.

8.1 Summary

In Chapter 3, a MILP for slack matching circuits composed entirely of a specified class of half buffers was derived. Each half buffer in a circuit is parametrized by a set of constants. The slack matching buffer is also parametrized by this set of constants, and must be a member of the specified class of half buffers. The MILP is formulated as a list of constraints, which when satisfied, ensure that a process graph with n_i slack matching buffers inserted on each channel i has cycle time at most τ_0. The MILP can be solved using standard MILP solvers. The results of slack matching circuits from the Lutonium [19] and MiniMIPS [18] are described. The cycle time of the fetch loop of the Lutonium was reduced from 68 transitions to 22 transitions. The MILP was generated in 0.1s and solved in 0.6s.

In Chapter 4, it was shown that the slack matching problem is NP-complete via a reduction from the subset sum problem.

In Chapter 5, a more general MILP for the slack matching problem was derived. This MILP can be used to slack match any circuit that can be modeled as a repetitive ER system. The only restriction is on the buffer used for slack matching. It must be possible to parametrize the critical paths between input and output transitions of the repetitive ER system of a pipeline of $n \geq 0$ buffers as a linear function of the variables that encode n. The results of slack matching an example circuit from the Lutonium [19] was described. This MILP was generated in 30s and solved in 2s. The reduction in cycle time observed was the same as for the MILP described in Chapter 3.

In Chapter 6, some simplifying assumptions are stated that allow a designer to slack match a circuit by hand. All constraints from in the MILP described in Chapter 3 fall under one of two simple rules that must be satisfied in order for a circuit to be slack matched.

In Chapter 7, the relationship between slack matching and other problems in circuit design is explored. First it is shown how slack matching may be integrated into an automated synthesis method for QDI circuits. Next slack matching QDI circuits is compared to the retiming problem in synchronous circuits.

8.2 Future Work

For the circuits tried so far, solving the MILPs derived in Chapter 3 and 5 has proven to be tractable. Further examples need to be tried in order to determine when solving the MILPs becomes intractable. When solving the MILP is intractable, heuristics for solving MILPs can be used to slack match circuits. Alternatively, it may be possible to slack match circuits hierarchically.

The proof of NP-completeness of slack matching relies on a reduction from subset sum. This problem is not NP-complete in the strong sense. That is there exists a pseudo-polynomial time algorithm that solves this problem. A pseudo-polynomial time algorithm is one that takes polynomial time in the encoding of the problem, however this encoding is exponentially larger than another encoding of the problem.

This suggests that slack matching may not be NP-complete in the strong sense. If this is indeed the case, then a pseudo-polynomial time algorithm would be of great use to the designer.

ER systems can only model disjunctions in PRS by considering specific scenarios. Further work is needed to develop models that allow a designer to estimate the cycle time of circuit over a range of different scenarios, rather than a specific scenario.

As the feature size of CMOS technology becomes smaller, it is becoming harder to predict the exact delays of circuit components. Delays of circuit components are being modeled as random variables instead. Recall that the expected value of a sum of random variables is the sum of the expected values of the random variables. Thus, the analysis in this thesis can be used to slack match a circuit so that the expected value of its cycle time is at most a specified target. Further work is needed in order to frame as a MILP, the problem of slack matching a circuit so that the expected value and variance of the circuits' cycle time are within specified bounds.

Appendix A

Notation

A.1 CHP

In this section, a brief summary of the CHP notation is provided. All variables are local to the process in which they appear. $x := a$ denotes the assignment of a to x. $I?x$ denotes an assignment of the value on the input port I to variable x. Similarly, $O!x$ assigns the value of x to output port O. These assignments to and from ports are blocking. 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 $*[\textbf{true} \rightarrow S]$ is abbreviated $*[S]$.

A.2 Handshaking Expansion(HSE)

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 $*[\textbf{true} \rightarrow S]$ is abbreviated

$*[S]$.

A.3 Production rule sets

QDI circuits can be described as production rule sets. A *production rule set(PRS)* is a set of production rules. A *production rule(PR)* is a guarded command of the form $G \rightarrow t$. G is the *guard* of the PR, and is a boolean expression; t is the *target* of the PR and is an assignment of the value **true** or **false** to a variable. The assignments of the values **true** and **false** to variable, v, are respectively denoted $v\uparrow$ and $v\downarrow$. Let $R(v\uparrow)$ be the predicate $v = \textbf{true}$. Similarly, let $R(v\downarrow)$ be the predicate $v = \textbf{false}$.

The execution of a PRS is an unbounded sequence of firings. A *firing* of the rule $G \rightarrow t$ with G **true** amounts to the execution of t, a firing with G **false** amounts to a **skip**. An *effective firing* is a firing that changes the value of a variable.

A valid PRS is stable and non-interfering. A PRS is *non-interfering* if for all pairs of rules of the form $(G_1 \rightarrow v\uparrow, G_2 \rightarrow v\downarrow)$, $\neg G_1 \vee \neg G_2$ is true at all times. A PRS is *stable* if for all rules $G \rightarrow t$, G changes from **true** to **false** only when $R(t)$ holds.

Bibliography

[1] *GNU Linear Programming Kit, Version 4.9.* http://www.gnu.org/software/glpk/glpk.html.

[2] P. Beerel, A.Lines, M.Davies, and N.H. Kim. Slack matching asynchronous designs. In *Proc. 12th IEEE International Symposium on Asynchronous Circuits and Systems*, March 2006.

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

[4] T.H. Cormen, C.E.Leiserson, and R.L.Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[5] Dash Optimization. *Xpress-MP.* http://www.dashoptimization.com/home/services/publicatio

[6] Kenneth Fazel, Lun Li, Mitch Thornton, Robert B. Reese, and Cherrice Traver. Performance enhancement in phased logic circuits using automatic slack-matching buffer insertion. In *GLSVLSI '04: Proceedings of the 14th ACM Great Lakes symposium on VLSI*, pages 413–416, New York, NY, USA, 2004. ACM.

[7] M.R. Garey and D.S. Johnson. *Computers and Intractability A Guide to the Theory of NP-Completeness.* Freeman and Company, 1979.

[8] M.R. Greenstreet and K. Steiglitz. Bubbles can make self-timed pipelines fast. *Journal of VLSI and Signal Processing*, 2(3):139–148, November 1990.

[9] S. Kim and P. Beerel. Pipeline optimization for asynchronous circuits: Complexity analysis and an efficient optimal algorithm. *IEEE Transactions on CAD*, 25(3):389–402, March 2006.

[10] T.K. Lee. *A General Approach to Performance Analysis and Optimization of Asynchronous Circuits*. PhD thesis, California Institute of Technology, 1995.

[11] C. Leiserson, F. Rose, and J. Saxe. Optimizing synchronous circuitry by retiming. In *Third Caltech Conference On VLSI*, March 1983.

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

[13] R. Manohar and A.J. Martin. Quasi-delay insensitive circuits are turing complete. In *Async96 Second International Symposium on Advanced Research in Asynchronous Circuits and Systems*, March 1996.

[14] 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.

[15] A.J. Martin. Compiling communicating processes into delay-insensitive vlsi-circuits. *Distributed Computing*, 1(4):226–234, 1986.

[16] A.J. Martin. The limitations to delay-insensitivity in asynchronous circuits. In *Sixth MIT Conference on Advanced Research in VLSI*. MIT Press, 1990.

[17] A.J. Martin. Synthesis of asynchronous circuits. Technical Report CS-TR-93-28, Caltech, 1993.

[18] 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.

[19] 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.

[20] A.J. Martin, M. Nyström, and P. Penzes. *Power-Aware Computing*, chapter ET^2: A Metric For Time and Energy Efficiency of Computation. Kluwer Academic Publishers, 2001.

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

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

[23] P. Prakash and A.J. Martin. Slack matching quasi delay insensitive circuits. In *Proc. 12th IEEE Intl Symposium on Asynchronous Circuits and Systems*, March 2006.

[24] J. Sparso and J. Staunstrup. Delay-insensitive multi-ring structures. *Integr. VLSI J.*, 15(3):313–340, 1993.

[25] S.C. Venkataramani, G.; Goldstein. Leveraging protocol knowledge in slack matching. *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*, pages 724–729, Nov. 2006.

[26] T. Williams. Latency and throughput tradeoffs in self-timed asynchronous pipelines and rings. Technical Report CSL-TR-90-431, Stanford, May 1990.

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