November 5, 1998

SRC Research Report 162

An Efficient Matching Algorithm for a High-Throughput, Low-Latency Data Switch

Thomas L. Rodeheffer
and
James B. Saxe

COMPAQ

Systems Research Center
130 Lytton Avenue
Palo Alto, CA 94301

http://www.research.digital.com/SRC/

An Efficient Matching Algorithm for a High-Throughput, Low-Latency Data Switch

Thomas L. Rodeheffer and James B. Saxe

November 5, 1998

Abstract

This paper focuses on two desired properties of cell-based switches for digital data networks: (1) data cells should not be detained inside the switch any longer than necessary (the *work-conserving* property) and (2) data cells that have been in the switch longer (older cells) should have priority over younger cells (the *order-conserving* property). A well-known, but expensive design of a work- and order-conserving switch is the output-queued switch.

A different switch design is the speedup crossbar switch, in which input buffers are connected to output buffers through a crossbar that runs at a multiple (called the *speedup*) of the external cell rate. A matching algorithm determines which cells are forwarded through the crossbar at any given time. Previous work has proposed a matching algorithm called the lowest output occupancy first algorithm (LOOFA). It is known that a LOOFA switch with speedup at least 2 is work-conserving.

We propose a refinement of LOOFA called the lowest output occupancy and timestamp first algorithm (LOOTFA). The main result of this paper is that a LOOTFA crossbar switch is work- and order-conserving provided that the speedup is at least 3. We prove this result and consider some generalizations.

Contents

1. Introduction

A *cell-based switch* processes fixed-sized chunks of data called *cells*, which arrive at switch *inputs*, pass through the switch proper, and depart from switch *outputs*. Each cell contains an identification of the single output to which it is destined. For convenience, we assume that the switch has the same number, N, of inputs and outputs and we assume that each input and output has the same capacity in cells per second. This capacity is called the *cell rate*, and its reciprocal, the *cell time*. We assume that all activities of the switch are synchronized to *slots*, each of which lasts one cell time. Figure 1 illustrates a cell-based switch.

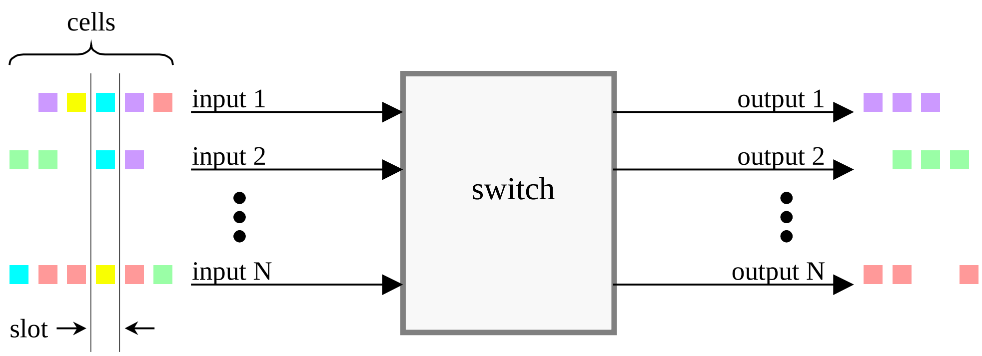

Figure 1: An $N \times N$ cell-based switch.

Although any realistic implementation would make extensive use of pipelining, for convenience we model the activity in the switch during each slot as a sequence of phases: an *inhale* phase, during which at most one cell from each input is accepted into the switch; a number of *transfer* phases, during which cells move around inside the switch; and an *exhale* phase, during which the switch emits at most one cell onto each output. See Figure 2. “Accepting” a cell during the inhale phase can be considered as the bookkeeping necessary to account for a cell that arrived during the previous slot, and “emitting” a cell during the exhale phase can be considered as the bookkeeping necessary to account for a cell that will depart during the following slot. These bookkeeping activities are covered by the pipeline delay and take no real time in an implementation.

The switch must contain buffer memory to hold temporary excesses of cells that result from short-term fluctuations in the arrival rate of cells destined to a given output. For example, multiple cells destined for the same output could be inhaled into the switch during the same slot, and the switch would have to hold these cells while the output exhaled them one by one. Mechanisms to prevent buffer overflow such as flow-control back-pressure or rate reservation are important but beyond the scope of this paper. We also ignore the rate- or phase-matching buffer at each input that is typically used to bring arriving cells into synchrony with the slot time of the switch.

In this paper we focus on two desired behaviors of a cell-based switch: (1) cells should not needlessly sit in buffers and (2) cells that have been in the switch longer (*older* cells) should have priority over younger cells.

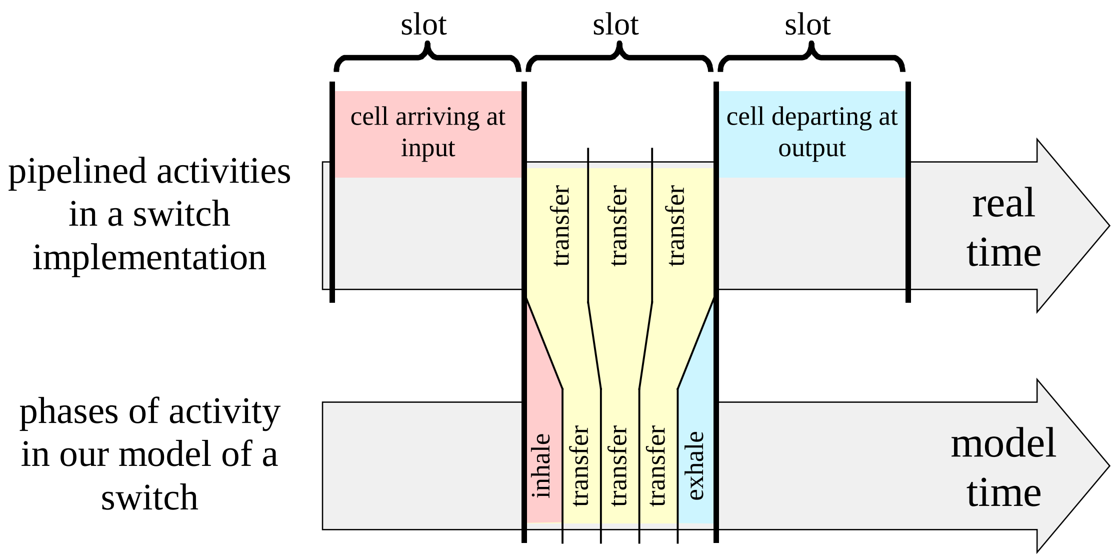

Figure 2: Model of the activities in a switch during a slot.

The *latency* of a cell is the number of slot boundaries between its inhale and its exhale. The first desired behavior can be stated formally as: the total latency over all cells is as small as possible. This is equivalent to the condition that each output always exhales some cell whenever there are any cells in the switch destined for that output. A switch that behaves in this manner is called *work-conserving*.

Whenever the switch contains multiple cells destined to the same output, the total latency is unaffected by the order in which the cells are exhaled. Given the choice, it seems good to give older cells priority over younger cells. Stated formally, we desire that each time an output exhales a cell, there are no older cells in the switch destined for that output. A switch that behaves in this manner is called *order-conserving*. In Section 5.3 we revisit the notion of "order-conserving" in a more general context.

A cell-based switch that is both work- and order-conserving should rightly be called *ideal*, but a more common term is the eponymous *output-queued*. To avoid confusion we refer to the behavior as *ideal* and the well-known implementation, described in the next paragraph, as *output-queued*.

The well-known implementation of an ideal cell-based switch is the *output-queued switch*, in which the switch takes cells directly into buffers local to each output, as shown in Figure 3. Assuming each non-empty output unit always exhales one of its oldest cells, this design is clearly work- and order-conserving, hence ideal. Unfortunately it also is expensive. Because all inputs could simultaneously inhale cells destined to the same output, the connection into each output unit must have a capacity of N times the cell rate: either N times wider (as in Figure 3), N times faster, or some combination. None of these alternatives scales well as N increases.

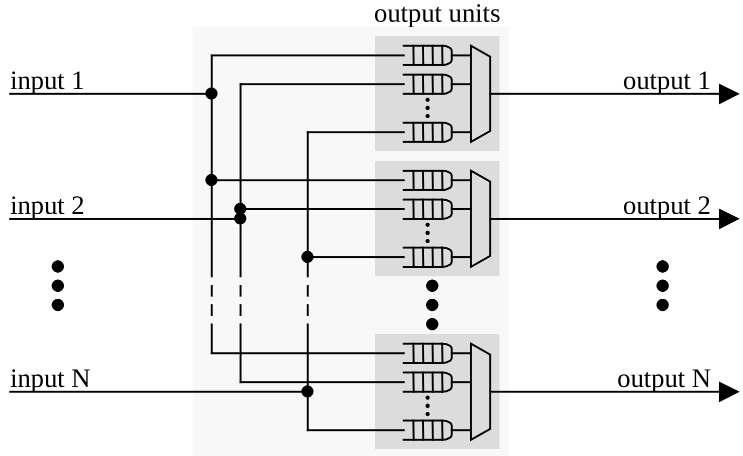

Figure 3: An N×N output-queued switch.

Another cell-based switch design is the *crossbar speedup switch*, which is illustrated in Figure 4. This switch contains input units, output units, and a crossbar interconnect. Cells are buffered at the input units and at the output units. The actions during each slot consist of an inhale phase, S (the *speedup*) transfer phases, and an exhale phase. During the inhale phase, each input unit inhales at most one cell and buffers it. During each transfer phase, the crossbar moves cells from input units to output units, subject to the restrictions that no more than one cell can be removed from any input unit and no more than one cell can be delivered to any output unit. During the exhale phase, each output unit removes at most one cell from its buffer and exhales it.

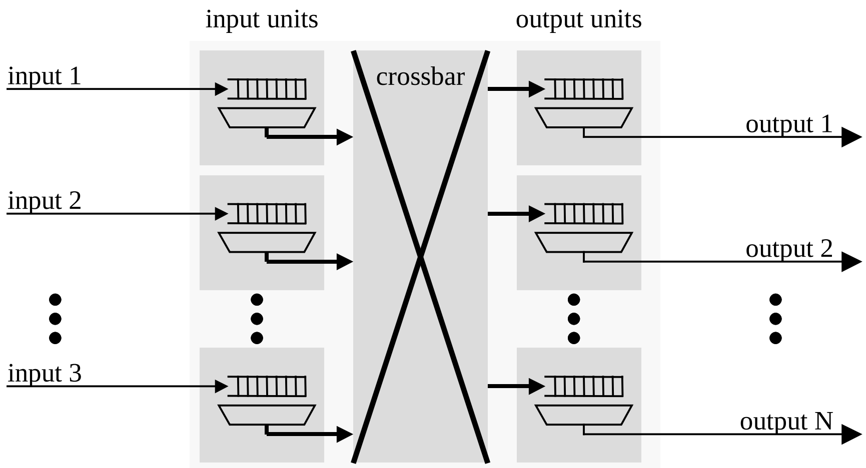

Figure 4: A crossbar speedup switch.

Since each connection between the crossbar and an input or output unit is required to transfer at most one cell per transfer phase, of which there are S per slot, each such connection requires a bandwidth of only S times the cell rate.

Each transfer phase proceeds in two parts: first a matching algorithm selects which cells in the input units to transfer (the *match*), and then the selected cells are transferred. We say that the cells in the input units *compete* for *inclusion* in the match. No pair of included cells can *conflict*, either by sharing the same input (which would be an *input conflict*) or sharing the same output (which would be an *output conflict*). The matching algorithm typically produces a *maximal match*, in which no additional cell can be included because each non-included cell has a conflict with some included cell. Since exactly the included cells are transferred, we also call them the *transferred* cells.

In the types of crossbar speedup switch we investigate, some *ordering* of cells is used to determine which cells are more *important* and thus win the competition. Different matching algorithms use different orderings.

Typically, each input unit buffers its cells in a separate queue for each output, as shown in Figure 5. Although illustrated as separate queues, a linked-list implementation is typical, and the usual name for these structures is *virtual output queues*. This design requires that the oldest cell in each queue always be a most important cell in that queue. Hence the oldest cell can always be included in a match in preference to any younger cell in its queue, and in fact the younger cells need not even be considered.

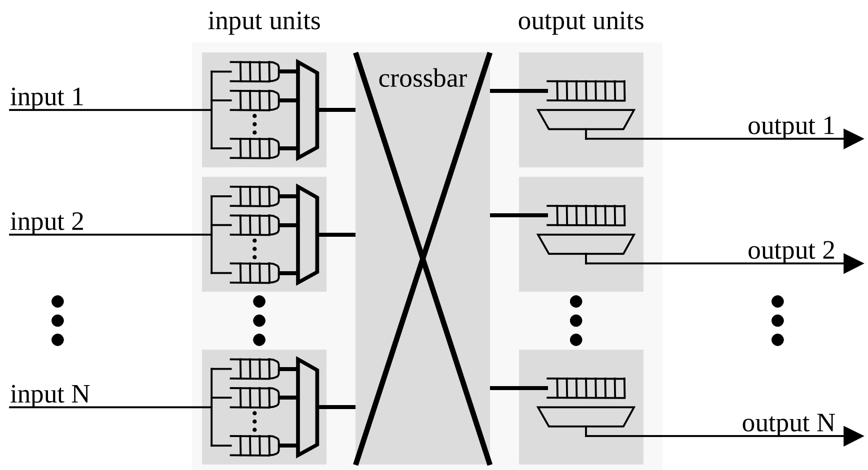

Figure 5: A crossbar speedup switch with (virtual) output queues.

If the matching algorithm can be designed so that for each output, some cell destined to that output (if any exist) is always present in the output unit at the beginning of the exhale phase, then the crossbar speedup switch will be work-conserving. Krishna *et al.* [1] have developed a matching algorithm called the *lowest output occupancy first algorithm* (LOOFA) that achieves this property provided that the speedup S is at least 2. The occupancy of an output is the number of cells currently buffered in the output unit. In LOOFA, a cell destined to an output with lower occupancy is more important than a cell destined to an output with higher occupancy. Intuitively, an output unit containing fewer cells will need another cell sooner than an output unit containing more cells and hence cells destined to the lower occupancy output should be more important.

If the matching algorithm can be designed so that for each output, an oldest cell destined to that output (if any exist) is always present in the output unit at the beginning of the exhale phase, then the crossbar speedup switch will be order-conserving in addition to being work-conserving—that is, it will be ideal. Prabhakar and McKeown [2] have developed a matching algorithm called the *most urgent cell first algorithm* (MUCFA) that achieves this property provided that the speedup S is at least 4. In their design, the switch schedules an exhale slot to each cell as it is inhaled, using the next available (not-yet-scheduled) exhale slot for the cell's destined output. Lower-numbered inputs get priority when the switch simultaneously inhales multiple cells destined to the same output. A cell's *urgency* is the number of slot boundaries remaining until its scheduled exhale. In MUCFA, a cell with lower urgency is more important than a cell with higher urgency. Clearly such a switch is ideal if it exhales each cell when its urgency is zero.

Both LOOFA and MUCFA use matching algorithms that guarantee that each non-included cell has a conflict with some included cell that is at least as important, according to their respective definitions of importance, as the non-included cell. As a consequence, their matches are maximal.

Since LOOFA takes no account of cells' ages, there is clearly no guarantee that it is order-conserving. However, the slight modification of resolving ties in output occupancy by favoring older cells produces an ideal switch provided that the speedup S is at least 3. We call this refinement the *lowest output occupancy and timestamp first algorithm* (LOOTFA). The fact that a LOOTFA switch with $S \geq 3$ is ideal is our main result.

2. Formal model of a crossbar speedup switch

In this section we present our notation and a formal model of a crossbar speedup switch. The formal model defines the state of the switch and the allowable changes in this state that can happen during each phase. In a LOOFTA switch, the matcher and the output selectors further constrain the behavior. In any specific execution history, the sequence of input data also constrains the behavior.

The formal model has two parameters, N and S:

N	the number of inputs of the switch; also the number of outputs
S	the crossbar speedup factor

2.1. Slot structure

Time is divided into slots. Each slot consists of an inhale phase, S transfer phases, and an exhale phase. We label phase boundaries with consecutive integers starting with 0. The phase beginning at boundary b is called phase b. See Figure 6.

In Section 5.2 we consider a more general phase arrangement.

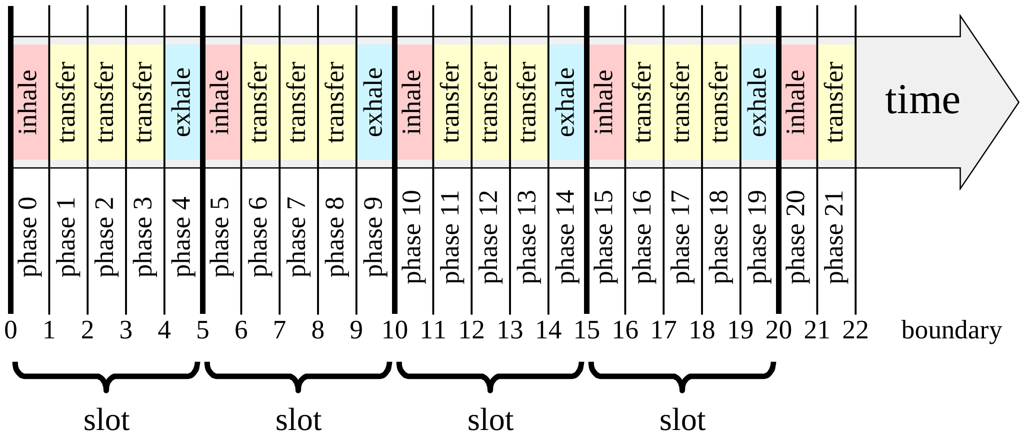

Figure 6: Example slot structure and phase boundary labels (*S*=3).

2.2. Basic notational conventions

We use the following notational conventions:

i	an input, $1 \le i \le N$
o	an output, $1 \le o \le N$
h	(the beginning of) an inhale phase
x	(the beginning of) a transfer phase
e	(the beginning of) an exhale phase
b	(the beginning of) any phase
c	a cell
$i(c)$	cell c's input
$o(c)$	cell c's destined output
$h(c)$	cell c's inhalation phase: c is inhaled during phase $h(c)$

2.3. State variables

The model has the following state variables:

IB_b	the set of cells in any input unit at time b
OB_b	the set of cells in any output unit at time b

2.4. Cell input or output subset notation

Given an arbitrary set C of cells, we use the following subscript notation for identifying subsets consisting of those cells with a given input or output (regardless of whether the cells are present in the switch at any given time):

$C_{\mathtt{i}=i} \equiv \{c \in C : i(c) = i\}$	those cells in C with input i
$C_{\mathtt{i}\neq i} \equiv \{c \in C : i(c) \neq i\}$	those cells in C not with input i
$C_{\mathtt{o}=o} \equiv \{c \in C : o(c) = o\}$	those cells in C destined to output o
$C_{\mathtt{o}\neq o} \equiv \{c \in C : o(c) \neq o\}$	those cells in C not destined to output o

Here are three examples of this notation:

$IB_{b,\mathtt{i}=i}$	$= \{c \in IB_b : i(c) = i\}$	cells in input unit *i* at time *b*
$IB_{b,\mathtt{i}=i,\mathtt{o}=o}$	$= \{c \in IB_b : i(c) = i \wedge o(c) = o\}$	cells in $IB_{b,\mathtt{i}=i}$ destined to output *o*
$OB_{b,\mathtt{o}=o}$	$= \{c \in OB_b : o(c) = o\}$	cells in output unit *o* at time *b*

2.5. Conflict notation

Cells that share an input or an output are in conflict and cannot both be transferred in the same phase. We use the following notation for the relation of two cells in conflict:

$c_1 \sim c_2$	$\equiv i(c_1) = i(c_2) \vee o(c_1) = o(c_2)$	input or output conflict

2.6. Cell ordering notation

We distinguish different cell orderings using subscripts:

$c_1 <_y c_2$	c_1 precedes (is more important than) c_2 according to ordering *y*
$c_1 <_z c_2$	c_1 precedes c_2 according to *z*
$c_1 =_z c_2$	c_1 ties c_2 according to *z*
$c_1 \leq_z c_2$	c_1 precedes or ties c_2 according to *z*

In all of the orderings we use in this paper, two cells *tie* if and only if neither precedes the other, and furthermore, as suggested by our notation, tying is an equivalence relation. We use the notation $<_{y,z}$ to designate the ordering derived from $<_y$ with ties broken by $<_z$:

$c_1 <_{y,z} c_2$	$\equiv c_1 <_y c_2 \vee (c_1 =_y c_2 \wedge c_1 <_z c_2)$	precedes according to *y* then *z*
$c_1 =_{y,z} c_2$	$\equiv c_1 =_y c_2 \wedge c_1 =_z c_2$	ties according to *y* then *z*

Next we give the initial state of the switch and the allowable changes in the state during inhale, transfer, and exhale phases.

2.7. The initial state

Initially there are no cells in the switch.

$\lvert IB_0 \rvert = 0$	the input buffer initially is empty
$\lvert OB_0 \rvert = 0$	the output buffer initially is empty

2.8. An inhale phase

For any inhale phase *b*, there exists a set of inhaled cells H_b such that:

$OB_{b+1} = OB_b$	the output buffer does not change
$IB_{b+1} = IB_b \cup H_b$	inhaled cells arrive in the input buffer
$\forall i : \lvert H_{b,\mathtt{i}=i} \rvert \le 1$	each input inhales at most one cell
$\forall c \in H_b : h(c) = b$	inhalation time is correct

2.9. A transfer phase

For any transfer phase *b*, there exists a set of transferred cells X_b such that:

$X_b \subseteq IB_b$	transfer a subset of the input buffer
$IB_{b+1} = IB_b - X_b$	transferred cells depart from the input buffer
$OB_{b+1} = OB_b \cup X_b$	transferred cells arrive in the output buffer
$\forall i : \lvert X_{b,\mathtt{i}=i} \rvert \le 1$	at most one transferred cell for each input
$\forall o : \lvert X_{b,\mathtt{o}=o} \rvert \le 1$	at most one transferred cell for each output

The set of transferred cells X_b is the set of cells included in the matching for phase *b*. In a LOOTFA switch, X_b also satisfies an additional condition given in Section 3.5.

2.10. An exhale phase

For any exhale phase *b*, there exists a set of exhaled cells E_b such that:

$IB_{b+1} = IB_b$	the input buffer does not change
$E_b \subseteq OB_b$	exhale a subset of the output buffer
$OB_{b+1} = OB_b - E_b$	exhaled cells depart from the output buffer
$\forall o : \lvert E_{b,\mathtt{o}=o} \rvert \le 1$	each output exhales at most one cell

In a LOOTFA switch, E_b also satisfies additional conditions given in Section 3.6.

3. The LOOTFA switch

In this section we present the additional conditions that a crossbar speedup switch must satisfy in order to be a LOOTFA switch and we develop concepts specific to the LOOTFA switch.

3.1. Output occupancy, oo_b

We define the *output occupancy* $oo_b(c)$ of a cell c at time b as the number of cells in c's destined output unit at time b. Formally,

$$oo_b(c) \equiv \left|OB_{b,\mathrm{o}=o(c)}\right|.$$

3.2. Output occupancy ordering, $<_{oo(b)}$

Given any two cells c_1, c_2, we say that c_1 precedes c_2 according to the *output occupancy ordering* at time b, written $c_1 <_{oo(b)} c_2$, iff at time b, the output occupancy of c_1 is less than the output occupancy of c_2. Formally,

$$c_1 <_{oo(b)} c_2 \equiv oo_b(c_1) < oo_b(c_2).$$

3.3. Timestamp ordering, $<_t$

Given any two cells c_1, c_2, we say that c_1 precedes c_2 according to the *timestamp ordering*, written $c_1 <_t c_2$, if and only if c_1 is inhaled before c_2. Formally,

$$c_1 <_t c_2 \equiv h(c_1) < h(c_2).$$

The timestamp ordering indicates which cells are older than others. In Section 5.3 we consider alternative definitions of the timestamp ordering.

3.4. Transfer time ordering, $<_x$

Given any two cells c_1, c_2, we say that c_1 precedes c_2 according to the basic transfer time ordering, written $c_1 <_{bx} c_2$, if and only if c_1 is transferred before c_2. We consider that a cell that is actually transferred is "transferred before" a cell that is never transferred. Formally,

$$c_1 <_{bx} c_2 \equiv \exists x_1 : c_1 \in X_{x_1} \wedge \left(\left(\exists x_2 : c_2 \in X_{x_2} \wedge x_1 < x_2\right) \vee \neg\left(\exists x_2 : c_2 \in X_{x_2}\right)\right).$$

We resolve ties in $<_{bx}$ arbitrarily to produce the total ordering $<_x$, called the *transfer time ordering*.

Note that the transfer time ordering is a property of an execution history of the switch, and is not in general available from the switch state at any moment in time. The transfer time is not used in the implementation of the switch, but only in our analysis of its behavior. We use $<_x$ in the definition of the *least important relevant cell* in Section 4.5. The oracular nature of $<_x$ enables us to pick the cell that an execution history in fact treats as less important in the event of a tie in the matching condition.

3.5. The LOOTFA matching condition and $w(b)$

Like LOOFA and MUCFA, in each transfer phase LOOTFA requires that each non-included cell have a conflict with some included cell that is at least as important. Roughly speaking, LOOTFA uses a definition of importance that favors cells with lower output occupancies, breaking ties in favor of cells with earlier timestamps.

A subtlety arises at this point. Whereas a cell's timestamp never changes, a cell's output occupancy can change over time. In particular, after any transfer phase, the relative output occupancies of the cells surviving in the input buffer may be different from what they were at the beginning of the phase. Since rapidly constructing a match is crucial to the performance of the switch, an implementation would most likely pipeline this process as much as possible. Reevaluating the relative importance of surviving cells on every transfer phase seems like it would be bothersome.

It turns out to be sufficient for the transfer phase to construct its matching based on output occupancies as they were at the end of the most recent inhale phase. This has the consequence that the relative importance of surviving cells does not change during the transfer phases in the same slot, which seems like a property that could be exploited in a pipelined implementation.

We define the function $w(b)$ of time b as the time at the end of the most recent inhale phase before b. Formally,

$$w(b) = \begin{cases} 0 & \text{if } b = 0 \\ b & \text{if phase } b-1 \text{ is an inhale phase} \\ w(b-1) & \text{otherwise} \end{cases}$$

In Section 5.1 we consider alternative definitions of w.

(Note that since the inhale phase does not affect output occupancies, we could equivalently use the "initial" output occupancies as of the beginning of the current slot. Krishna *et al.* [1] discovered that all of the transfer phases in the same slot could use initial output occupancies when they proved that an $S \geq 2$ LOOFA switch was work-conserving.)

Now we can define the LOOTFA *matching condition*. For every transfer phase b, a LOOTFA switch satisfies the following condition in addition to the transfer phase conditions in Section 2.9:

$$\forall c \in IB_b - X_b : \exists c' \in X_b : c' \sim c \wedge c' \leq_{oo(w(b)),t} c \,.$$

That is, for each cell c in the input buffer that is not included in the match, there exists some conflicting, included cell c' that is at least as important as c, where a cell is more important than another if it has a lower output occupancy at time $w(b)$ or, in the event of a tie, if it has an earlier timestamp. Since c' is transferred while c remains in the input buffer, we necessarily have $c' <_{oo(w(b)),t,x} c$. We say that c' is transferred *in preference to* *c*.

3.6. The LOOTFA exhale conditions

For every exhale phase *b*, a LOOTFA switch satisfies the following conditions in addition to the exhale phase conditions in Section 2.10:

$$\forall o : \left|OB_{b,\mathrm{o}=o}\right| > 0 \Rightarrow \left|E_{b,\mathrm{o}=o}\right| > 0 \qquad \textit{OB} \text{ work-conserving}$$
$$\forall c \in E_b : \forall c' \in OB_{b,\mathrm{o}=o(c)} : c \leq_t c' \qquad \textit{OB} \text{ order-conserving}$$

That is, each non-empty output *o* always exhales a cell, and the cell it exhales precedes or ties according to the timestamp ordering all cells in the output buffer destined to *o*.

4. The LOOTFA theorem

We now come to our main result.

Theorem (LOOTFA): A LOOTFA switch with speedup $S \geq 3$ is ideal.

The rest of Section 4 is devoted to a proof of this theorem. We assume an execution history that is a counterexample, define a number of attributes (*e*, *fc*, R_b, *h*, $lirc_b$, OBT_b, p_b, **H**, **X**, and **E**) of this execution history, and finally arrive at a contradiction.

4.1. Earliest failing exhale phase, *e*

Recall from Section 1 that a switch is ideal if and only if it is both work-conserving and order-conserving. To be work-conserving, the switch must ensure that whenever there are any cells in the switch destined to output *o* at the beginning of an exhale phase *b*, output *o* exhales some cell during phase *b*. To be order-conserving, the switch must ensure that whenever an output *o* exhales some cell *c*, there are no cells in the switch destined to output *o* that precede *c* according to the timestamp ordering.

Formally, a switch is ideal if, in every execution history, the following conditions both hold for every exhale phase *b*:

$$\forall o : \left|\left(IB_b \cup OB_b\right)_{\mathrm{o}=o}\right| > 0 \Rightarrow \left|E_{b,\mathrm{o}=o}\right| > 0 \qquad \text{work-conserving}$$
$$\forall c \in E_b : \forall c' \in \left(IB_b \cup OB_b\right)_{\mathrm{o}=o(c)} : c \leq_t c' \qquad \text{order-conserving}$$

We say that an exhale phase *fails* if it violates one or both of the above conditions. (For example, if at the beginning of an exhale phase *b*, a crossbar speedup switch has a cell destined to *o* in its input buffer but no cells destined to *o* in its output buffer, then exhale phase *b* is sure to fail.)

In our assumed counterexample execution history, there must be some exhale phase that fails. We define *e* to be the earliest such failing exhale phase.

4.2. The failing cell, *fc*

In order for exhale phase *e* to fail, there must be some cell $c \in \left(IB_e \cup OB_e\right)$ in the switch such that either (1) no cell is exhaled on output $o(c)$ (which would violate work-conserving) or (2) a cell is exhaled on output $o(c)$ that c precedes according to the timestamp ordering (which would violate order-conserving). We pick one such cell and call it fc, the *failing* cell.

We claim that at time *e*, cell *fc* must be in the input buffer and it must precede all cells in output $o(fc)$ according to the timestamp ordering. This claim follows from the LOOTFA exhale conditions of Section 3.6.

Formally, we first prove that $\forall c \in OB_{e,o=o(fc)} : fc <_t c$. Assuming the contrary, there exists a cell $c \in OB_{e,o=o(fc)}$ such that $c \le_t fc$. Then from the *OB* work-conserving condition, output $o(fc)$ must exhale some cell c', and from the *OB* order-conserving condition, we have $c' \le_t c$, whence $c' \le_t fc$. This contradicts the definition of *fc*, so our statement is proved.

Since *fc* does not precede itself according to the timestamp ordering, *fc* cannot be in $OB_{e,o=o(fc)}$, and therefore $fc \notin OB_e$. By definition $fc \in (IB_e \cup OB_e)$, so we have $fc \in IB_e$. This completes the proof of our claim.

In summary, we have

$fc \in IB_e$, and

$\forall c \in OB_{e,o=o(fc)} : fc <_t c$.

The rest of the proof proceeds as follows. We define a set of *relevant* cells, which are those cells sharing the same input as fc that contribute to allowing fc to survive in the input buffer until the earliest failing phase *e*. We define the *least important relevant cell* at time *b* and prove a property of its output occupancy. We examine the *output buffer trailing* cells, which are those cells in the output unit $o(fc)$ that are preceded by fc according to the timestamp ordering. Then we define a *potential* at time *b* as a linear combination of various salient quantities in the switch state at time *b*. We establish a lower bound on the potential at the inhalation of the first relevant cell, push this bound forward phase by phase, and thus obtain a lower bound at time *e*. Finally we directly compute the potential at time *e* and obtain a value that violates the lower bound, thus showing a contradiction.

4.3. Relevant cells, *R*

We define a cell *c* to be *relevant* if:

(1) $c = fc$ or

(2) *c* shares the same input as fc and is transferred in preference to some relevant cell during some transfer phase $b < e$.

Recall from Section 3.5 that a cell *c* is said to be transferred "in preference to" a cell c' during transfer phase *b* if and only if *c* is transferred, c' survives in the input buffer, *c* and c' conflict, and *c* is at least as important as c'; formally,

$$c \in X_b \wedge c' \in IB_b - X_b \wedge c \sim c' \wedge c \le_{oo(w(b)),t} c'.$$

Intuitively, the relevant cells are fc and cells that, directly or indirectly, delay the transfer of fc by means of input conflicts.

We define R as the set of all relevant cells. For any time b we define R_b as the set all of relevant cells present in the input buffer at time b. Formally,

$$R_b \equiv R \cap IB_b .$$

A transfer phase during which some relevant cell is transferred we call an *R-transfer phase*. A transfer phase during which no relevant cell is transferred we call a *nonR-transfer phase*.

4.4. Earliest inhale of a relevant cell, *h*

Each relevant cell $c \in R$ has an inhalation phase $h(c)$. We define h to be the earliest inhalation phase of any relevant cell. Formally,

$$h \equiv \min_{c \in R} h(c).$$

Since R is non-empty ($fc \in R$), h is well-defined.

We claim that for any time b in the range $h < b \le e$, we have $|R_b| > 0$. Clearly $|R_{h+1}| > 0$, since the switch has just inhaled a relevant cell and has not yet had a chance to transfer it. An R-transfer phase $b < e$ transfers a relevant cell $c \in R_b$, but since c cannot be fc (because fc is not transferred before e), c must be transferred in preference to some other relevant cell $c' \in R_b$, and consequently we have $c' \in R_{b+1}$. No other phase can remove a relevant cell from the input buffer, so the claim is proved.

4.5. Least important relevant cell, *lirc*$_b$

For any time b in the range $h < b \le e$, we define the *least important relevant cell* $lirc_b$ at time b as the maximum element of R_b according to $<_{oo(w(b)),t,x}$. That is,

$$lirc_b \in R_b \wedge \forall c \in R_b : c \le_{oo(w(b)),t,x} lirc_b .$$

Since $|R_b| > 0$ and $<_x$ is total, the least important relevant cell exists and is unique. Note that the least important relevant cell is defined in terms of the output occupancy ordering as it is at time $w(b)$, which, not surprisingly, is the output occupancy ordering used in the LOOTFA matching condition.

We now prove two useful lemmas about the least important relevant cell. Note that these lemmas relate to the assumed counterexample execution history with respect to which e, R_b, h, and $lirc_b$ are defined.

Lemma (*lirc* survival): For any phase b in the range $h < b < e$, we have $lirc_b \in R_{b+1}$.

Proof: By definition $lirc_b \in R_b$. If *b* is an inhale phase, an exhale phase, or a transfer phase that does not transfer $lirc_b$, then $lirc_b$ survives in the input buffer at time $b+1$, and consequently $lirc_b \in R_{b+1}$. It remains to consider the case in which *b* is a transfer phase and $lirc_b \in X_b$. In this case, we must have $lirc_b \neq fc$, since fc is not transferred before *e*. From the definition of relevance, $lirc_b$ must be transferred in preference to some other relevant cell $c \in R_b$, which means that $lirc_b$ is at least as important as *c*, that is $lirc_b \leq_{oo(w(b)),t} c$. Since $lirc_b$ is transferred before *c*, we have $lirc_b <_x c$. But this gives us $lirc_b <_{oo(w(b)),t,x} c$, which contradicts the definition of $lirc_b$. This completes the proof.

Lemma (*lirc* output occupancy): For any phase *b* in the range $h < b < e$, we have

$$oo_{b+1}(lirc_{b+1}) \geq oo_{b+1}(lirc_b).$$

Proof: Intuitively, either the choice of $lirc_{b+1}$ is based on output occupancies at time $b+1$ or else $lirc_{b+1} = lirc_b$. By definition, $lirc_{b+1}$ is the maximum element of R_{b+1} under $<_{oo(w(b+1)),t,x}$. Since we have $lirc_b \in R_{b+1}$ by the previous lemma, it follows that $lirc_b \leq_{oo(w(b+1)),t,x} lirc_{b+1}$ and hence $lirc_b \leq_{oo(w(b+1))} lirc_{b+1}$. If $w(b+1) = b+1$ then we are done. Otherwise, by the definition of *w* (see Section 3.5), $w(b+1) = w(b)$ and phase *b* cannot inhale any cells. Since $lirc_{b+1}$ cannot have been inhaled during phase *b*, it must have been in the input buffer at time *b*, and consequently $lirc_{b+1} \in R_b$. By definition, $lirc_b$ is the maximum element of R_b under $<_{oo(w(b)),t,x}$, so it follows that $lirc_{b+1} \leq_{oo(w(b)),t,x} lirc_b$. But $w(b+1) = w(b)$, so we have $lirc_{b+1} \leq_{oo(w(b+1)),t,x} lirc_b$. We now have $lirc_b$ and $lirc_{b+1}$ each at least as important as the other according to $<_{oo(w(b+1)),t,x}$. Since this ordering is total, it follows that $lirc_{b+1} = lirc_b$ and we are done.

4.6. Output buffer trailing cells, *OBT$_b$*

For any time *b* in the range $h < b \leq e$, we define the *output buffer trailing* cells OBT_b at time *b* as the set of those cells in output unit $o(fc)$ that are preceded by fc according to the timestamp ordering. Formally,

$$OBT_b \equiv \{ c \in OB_{b,o=o(fc)} : fc <_t c \}.$$

4.7. Potential, *p$_b$*

For any time *b* in the range $h < b \leq e$, we define the *potential* p_b at time *b* by the following magic formula:

$$p_b \equiv oo_b(lirc_b) - |OBT_b| - 2 \cdot |R_b|.$$

We establish a lower bound on the potential at time $h+1$, analyze the changes in potential with each phase, and show that the resulting lower bound on potential at time *e* contradicts the actual potential at time *e*.

4.8. Lower bound on potential at time *h*+1

To bound the potential at time $h+1$ we bound the components in its definition.

$oo_{h+1}(lirc_{h+1}) \geq 0$ An output occupancy cannot be negative.

$|OBT_{h+1}| = 0$ Consider any cell c in output unit $o(fc)$ at time $h+1$, that is, $c \in OB_{h+1,\text{o}=o(fc)}$. Cell c must be transferred during some earlier transfer phase $x(c) < h+1$, and since phase *h* is an inhale phase, we have $x(c) < h$. Cell c must be inhaled before it is transferred, hence $h(c) < x(c) < h$. Since *h* is the inhalation time of the earliest relevant cell and fc is relevant, we have $h \leq h(fc)$ and thus $h(c) < x(c) < h(fc)$. Hence from the definition of the timestamp ordering we have $c \leq_t fc$. So *c* is not an output buffer trailing cell.

$|R_{h+1}| = 1$ At time $h+1$ the switch has just inhaled the earliest relevant cell.

Combining the components, we have

$$\begin{aligned} p_{h+1} &= oo_{h+1}(lirc_{h+1}) - |OBT_{h+1}| - 2 \cdot |R_{h+1}| \\ &\geq -2. \end{aligned}$$

Next we consider the effects of each phase as *b* advances from $h+1$ to *e*.

4.9. Effect of an inhale phase

To bound the change in potential during an inhale phase *b*, we bound the changes of the components.

$oo_{b+1}(lirc_{b+1}) \geq oo_b(lirc_b)$

The output buffer is unchanged by an inhale phase, so we have $oo_{b+1}(lirc_b) = oo_b(lirc_b)$. Combining this with the lirc output occupancy lemma (Section 4.5) we get $oo_{b+1}(lirc_{b+1}) \geq oo_{b+1}(lirc_b) = oo_b(lirc_b)$.

$|OBT_{b+1}| = |OBT_b|$ The output buffer is unchanged by an inhale phase.

$|R_{b+1}| \leq |R_b| + 1$ Input $i(fc)$ can inhale at most one cell.

Combining the components, we have

$$p_{b+1} = oo_{b+1}(lirc_{b+1}) - |OBT_{b+1}| - 2 \cdot |R_{b+1}|$$

$$\geq oo_b(lirc_b) - |OBT_b| - 2 \cdot (|R_b| + 1)$$

$$= oo_b(lirc_b) - |OBT_b| - 2 \cdot |R_b| - 2$$

$$= p_b - 2.$$

4.10. Effect of an R-transfer phase

To bound the change in potential during an R-transfer phase *b*, we bound the changes of the components.

$oo_{b+1}(lirc_{b+1}) \geq oo_b(lirc_b)$

No cells can depart the output buffer, so $oo_{b+1}(lirc_b) \geq oo_b(lirc_b)$. Combining this with the lirc output occupancy lemma (Section 4.5) we get $oo_{b+1}(lirc_{b+1}) \geq oo_{b+1}(lirc_b) \geq oo_b(lirc_b)$.

$|OBT_{b+1}| \leq |OBT_b| + 1$

There might be a new output buffer trailing cell, but there can be at most one.

$|R_{b+1}| = |R_b| - 1$ Exactly one relevant cell is transferred.

Combining the components, we have

$$p_{b+1} = oo_{b+1}(lirc_{b+1}) - |OBT_{b+1}| - 2 \cdot |R_{b+1}|$$

$$\geq oo_b(lirc_b) - (|OBT_b| + 1) - 2 \cdot (|R_b| - 1)$$

$$= oo_b(lirc_b) - |OBT_b| - 2 \cdot |R_b| + 1$$

$$= p_b + 1.$$

4.11. Effect of a nonR-transfer phase

To bound the change in potential during a nonR-transfer phase *b*, we bound the changes of the components.

$oo_{b+1}(lirc_{b+1}) \geq oo_b(lirc_b) + 1$

Since $lirc_b$ is relevant, $lirc_b$ is not transferred during phase *b*. Therefore from the LOOTFA matching condition (Section 3.5) there must be some cell transferred in preference to $lirc_b$. Since any cell transferred in preference to $lirc_b$ and sharing input $i(lirc_b) = i(fc)$ would by definition be relevant, and since no relevant cell is transferred during a nonR-transfer phase, there must be some cell transferred in preference to $lirc_b$ that shares output $o(lirc_b)$. Therefore $oo_{b+1}(lirc_b) = oo_b(lirc_b) + 1$. Combining this with the lirc output occupancy lemma (Section 4.5) we get $oo_{b+1}(lirc_{b+1}) \geq oo_{b+1}(lirc_b) = oo_b(lirc_b) + 1$.

$|OBT_{b+1}| = |OBT_b|$ Since *fc* is relevant, *fc* is not transferred during phase *b*. Therefore from the LOOTFA matching condition (Section 3.5) there must be some cell transferred in preference to *fc*. Since any cell transferred in preference to *fc* and sharing input $i(fc)$ would by definition be relevant, and since no relevant cell is transferred during a nonR-transfer phase, there must be some cell transferred in preference to *fc* that shares output $o(fc)$. Let *c* be such a cell. Since *c* is transferred in preference to *fc*, we have $c \leq_{oo(w(b)),t} fc$. Since $o(c) = o(fc)$, we have $c =_{oo(w(b))} fc$ and hence $c \leq_t fc$. Therefore *c* is not an output buffer trailing cell. Since at most one cell can be transferred to any given output during a single transfer phase, *c* is the only cell transferred to output $o(fc)$ during phase *b*. So no output buffer trailing cells are transferred during phase *b*.

$|R_{b+1}| = |R_b|$ No relevant cell is transferred.

Combining the components, we have

$$
\begin{aligned}
p_{b+1} &= oo_{b+1}(lirc_{b+1}) - |OBT_{b+1}| - 2 \cdot |R_{b+1}| \\
&\geq (oo_b(lirc_b) + 1) - |OBT_b| - 2 \cdot |R_b| \\
&= oo_b(lirc_b) - |OBT_b| - 2 \cdot |R_b| + 1 \\
&= p_b + 1.
\end{aligned}
$$

4.12. Effect of an exhale phase

To bound the change in potential during an exhale phase *b*, we bound the changes of the components.

$oo_{b+1}(lirc_{b+1}) \geq oo_b(lirc_b) - 1$

Since output $o(lirc_b)$ can exhale at most one cell, we have $oo_{b+1}(lirc_b) \geq oo_b(lirc_b) - 1$. Combining this with the lirc output occupancy lemma (Section 4.5) we get $oo_{b+1}(lirc_{b+1}) \geq oo_{b+1}(lirc_b) \geq oo_b(lirc_b) - 1$.

$|OBT_{b+1}| = |OBT_b|$ Since $b < e$ and exhale phase *e* is assumed to be the earliest phase in which the switch fails, output $o(fc)$ cannot exhale any member of OBT_b.

$|R_{b+1}| = |R_b|$ The input buffer is unchanged.

Combining the components, we have

$$\begin{aligned} p_{b+1} &= oo_{b+1}(lirc_{b+1}) - |OBT_{b+1}| - 2\cdot|R_{b+1}| \\ &\ge (oo_b(lirc_b) - 1) - |OBT_b| - 2\cdot|R_b| \\ &= oo_b(lirc_b) - |OBT_b| - 2\cdot|R_b| - 1 \\ &= p_b - 1. \end{aligned}$$

4.13. Lower bound on potential at time *e*

To summarize the preceding sections, the effect on p_b of the phases between time $h+1$ and time *e* is as follows: each inhale phase decreases p_b by at most 2, each transfer phase (regardless of whether R-transfer or nonR-transfer) increases p_b by at least 1, and each exhale phase decreases p_b by at most 1. Let

$\mathbf{H} \equiv$ the number of inhale phases between time $h+1$ and time *e*,
$\mathbf{X} \equiv$ the number of transfer phases between time $h+1$ and time *e*, and
$\mathbf{E} \equiv$ the number of exhale phases between time $h+1$ and time *e*.

(Note that $\mathbf{E}$ does not include the failing phase, which starts at time *e*.) Then formally,

$$p_e \ge p_{h+1} - 2\cdot\mathbf{H} + \mathbf{X} - \mathbf{E}$$

Observe from the slot structure (Section 2.1) that in any interval starting at the end of an inhale phase and ending at the start of an exhale phase, there are more transfer phases than twice the number of inhale phases plus the number of exhale phases. In particular, we have

$$\mathbf{X} > 2\cdot\mathbf{H} + \mathbf{E}.$$

Combining the above with the lower bound $p_{h+1} \ge -2$ (Section 4.8), we have

$$\begin{aligned} p_e &\ge p_{h+1} - 2\cdot\mathbf{H} + \mathbf{X} - \mathbf{E} \\ &> p_{h+1} \\ &\ge -2. \end{aligned}$$

4.14. The potential at time *e*

Now let us compute p_e directly.

$$oo_e(lirc_e) = oo_e(fc)$$

From the definition of *R*, each relevant cell except *fc* is transferred during some transfer phase $b < e$. Hence we have $R_e = \{\ fc\ \}$ and thus $lirc_e = fc$.

$|OBT_e| = oo_e(fc)$ Since by assumption exhale phase e fails, all cells in output unit $o(fc)$ at time e must be preceded by fc according to $<_{oo(w(e)),t}$, that is, they are all output buffer trailing cells.

$|R_e| = 1$ $R_e = \{fc\}$.

Combining the components, we have

$$
\begin{aligned}
p_e &= oo_e(lirc_e) - |OBT_e| - 2 \cdot |R_e| \\
&= oo_e(fc) - oo_e(fc) - 2 \cdot 1 \\
&= -2,
\end{aligned}
$$

which contradicts the lower bound obtained in Section 4.13. Hence the assumption of a counterexample arrives at a contradiction, and the LOOTFA theorem is proved.

5. Generalizations

For clarity, we have presented LOOTFA with a number of concrete assumptions that are not strictly required by our proof. In this section we show some generalizations.

5.1. Generalized time of evaluation, *w*(*b*)

When transfer phase b decides which cells to include in the match, it evaluates the importance of cells based on output occupancies as of time $w(b)$. Our original definition of w (Section 3.5) causes each transfer phase to construct its match based on output occupancies as they are at the end of the most recent inhale phase. However, the only place in which we use properties of w is the proof of the *lirc* output occupancy lemma in Section 4.5, which succeeds if w satisfies the following conditions for each phase b:

$w(b+1) = w(b) \vee w(b+1) = b+1$ same as previous, or become current

$|IB_{b+1} - IB_b| > 0 \Rightarrow w(b+1) = b+1$ become current if any inhaled cells

The basic idea is that the switch maintains a record of output occupancies based on which it constructs the match for a transfer phase and, from time to time, the switch updates this record to the current output occupancies. A LOOTFA switch must update its output occupancies at the end of every inhale phase that actually inhales a cell, and it can also update whenever convenient.

For example, consider the definition $w(b) \equiv b$, which satisfies the generalized conditions. Under this definition, every transfer phase constructs its match based on current output occupancies. As explained in Section 3.5, this would likely be bothersome to implement. However, the resulting switch would be ideal provided $S \geq 3$.

5.2. Generalized phase arrangement

Our original phase arrangement was a sequence of slots each consisting of an inhale phase, S transfer phases, and an exhale phase (Section 2.1). However, the only place in which we use properties of the phase arrangement is the counting argument in Section 4.13. This argument requires that

$$\mathbf{X} > 2 \cdot \mathbf{H} + \mathbf{E},$$

for any interval starting at the end of an inhale phase and ending at the start of an exhale phase, where

$\mathbf{H} \equiv$ the number of inhale phases in the interval,
$\mathbf{X} \equiv$ the number of transfer phases in the interval, and
$\mathbf{E} \equiv$ the number of exhale phases in the interval.

In addition to our original slot structure, there are many other ways of arranging phases that satisfy this condition. For example, we can arrange the phases into *multislots*, in which each phase of a slot repeats n times, as illustrated in Figure 7.

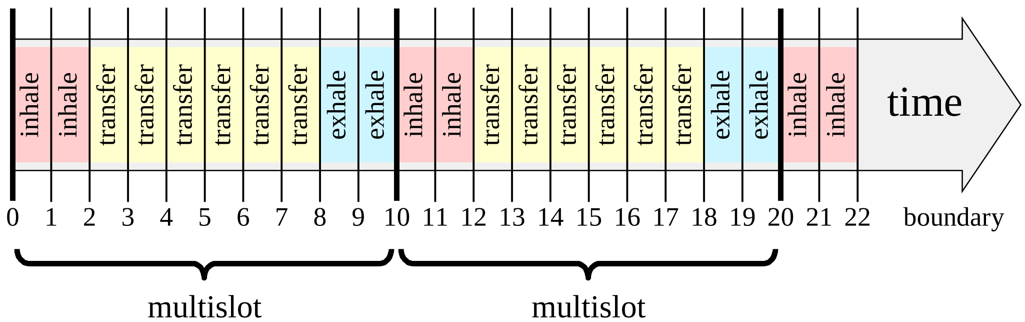

Figure 7: Example multislot structure (*n*=2, *S*=3).

Since all of the transfer phases in the same multislot are allowed to use the same output occupancies, a multislot implementation could probably be pipelined more extensively than a single slot design. This would achieve higher throughput at the cost of higher latency in real time due to increased discrepancy between real time and model time.

5.3. Generalized timestamp ordering, $<_t$

Our original timestamp ordering given in Section 3.3 said that a cell inhaled before another must precede the other according to the timestamp ordering. However, the only place in which we use properties of the timestamp ordering is the lower bound calculation in Section 4.8, which succeeds if $<_t$ satisfies the following *timestamp condition* for any cells c_1, c_2 and any transfer phase x:

$$h(c_1) < x < h(c_2) \quad \Rightarrow \quad c_1 \leq_t c_2 .$$

That is, if the inhalation phases of two cells are separated by an intervening transfer phase, the earlier cell must precede or tie the later cell according to the timestamp ordering.

The timestamp controls how a LOOTFA switch orders cells destined to the same output. Provided that the speedup $S \geq 3$, during each exhale phase each output exhales a cell that precedes or ties, according to the timestamp order, all cells in the switch destined to that output (if any). That is, for every exhale phase *b*, we have

$$\forall o : \left| \left(IB_b \cup OB_b \right)_{\mathrm{o}=o} \right| > 0 \Rightarrow \left| E_{b,\mathrm{o}=o} \right| > 0 \qquad \text{work-conserving}$$

$$\forall c \in E_b : \forall c' \in \left(IB_b \cup OB_b \right)_{\mathrm{o}=o(c)} : c \leq_t c' \qquad \text{“order-conserving”}$$

We could say that the timestamp ordering defines the meaning of “older” and “younger” and therefore the meaning of “order-conserving”. Perhaps it would be better to call the condition *timestamp-conserving*.

For example, our original timestamp ordering orders cells according to their inhalation phases. An output of an $S \geq 3$ LOOTFA switch using this timestamp exhales cells in order of inhalation phase, but an arbitrary order applies to cells inhaled during the same phase.

For a second example, consider the timestamp ordering which orders cells according to their inhalation phases and breaks ties by ordering cells according to their input number. Since an input can inhale at most one cell per phase, this is a total order. An output of an $S \geq 3$ LOOTFA switch using this timestamp exhales cells strictly according to this total order.

For a third example, consider a multislot switch (Section 5.2) in which the timestamp ordering orders cells according to their inhalation multislot, but arbitrarily orders cells that are inhaled during the repeated inhale phases of the same multislot. An output of an $S \geq 3$ LOOTFA switch using this timestamp exhales cells according to the timestamp ordering, which may or may not be useful.

For a fourth example, consider the trivial timestamp ordering, in which all cells tie. The effect of such a definition is to remove from the LOOTFA switch all consideration of the age of a cell, thus reducing it to a LOOFA switch. In this case the order-conserving success condition is trivially satisfied and the only interesting property is work-conserving. We can slightly modify our proof of the LOOTFA theorem to obtain a proof of the fact that an $S \geq 2$ LOOFA switch is work-conserving. Assuming a failing exhale phase *e*, using the same definitions of *fc*, R_b, *h*, $lirc_b$, $\mathbf{H}$, $\mathbf{X}$, and $\mathbf{E}$, and using the modified potential p_b at time *b* defined by

$$p_b \equiv oo_b\left(lirc_b\right) - \left|R_b\right|,$$

we can show that

$$p_{h+1} \geq -1,$$

$p_{b+1} \geq p_b - 1$ for each inhale phase *b*,

$p_{b+1} \geq p_b + 1$ for each transfer phase *b*,

$p_{b+1} \geq p_b - 1$ for each exhale phase *b*,

$p_e \geq p_{h+1} - \mathbf{H} + \mathbf{X} - \mathbf{E} > p_{h+1}$, and

$$p_e = -1,$$

which is a contradiction, hence proving that there can be no exhale phase that fails. (Note that under a timestamp ordering in which all cells tie, there are never any output buffer trailing cells.) Krishna *et al.* [1] proved that an $S \geq 2$ LOOFA switch is work-conserving. Our result here provides an alternate proof.

6. Computing a LOOTFA match

The obvious algorithm to compute a LOOTFA match is to repeatedly pick the most important non-conflicted cell until no more cells can be picked. We call this the *global minimum greedy* algorithm. It turns out that if the timestamp is the same for all cells (as in the fourth example of Section 5.3), then the match can also be computed by visiting the input units in an arbitrary order picking the most important non-conflicted cell in each.

6.1. The global minimum greedy algorithm

Given two sets C_1 and C_2 of cells, we define the set $C_1 /\!\sim C_2$ as those cells in C_1 that do not conflict with any cell in C_2. Formally,

$$C_1 /\!\sim C_2 \quad \equiv \{c_1 \in C_1 : \neg\exists c_2 \in C_2 : c_1 \sim c_2\}.$$

The *global minimum greedy* algorithm computes a LOOTFA match at time *b* with an iterative sequence $X_{b,0}$, $X_{b,1}$, …, $X_{b,Z}$ starting with $X_{b,0} = \{\,\}$ and producing $X_{b,z+1}$ from $X_{b,z}$ by adding a most-important (that is, minimal) element of $IB_b /\!\sim X_{b,z}$ according to $<_{oo(w(b)),t}$. There may be ties, in which case the choice between the tied minimal elements is arbitrary. When $IB_b /\!\sim X_{b,z}$ is empty, we declare that $Z = z$ and the algorithm is done. The result X_b is $X_{b,Z}$. Each step is called a *round*. Observe that $Z \leq N$.

Clearly any result of the global minimum greedy algorithm satisfies the formal model transfer phase requirements (Section 2.9),

$$X_b \subseteq IB_b,$$
$$\forall i : \left|X_{b,\mathrm{i}=i}\right| \leq 1, \text{ and}$$
$$\forall o : \left|X_{b,\mathrm{o}=o}\right| \leq 1.$$

To show that the result satisfies the LOOTFA matching condition (Section 3.5),

$$\forall c \in IB_b - X_b : \exists c' \in X_b : c' \sim c \wedge c' \leq_{oo(w(b)),t} c,$$

we introduce the following invariant, which we claim holds at any round *z*:

$$\forall c \in IB_b : \left(\exists c' \in X_{b,z} : c' \sim c \wedge c' \leq_{oo(w(b)),t} c\right) \vee \left(c \in IB_b /\!\sim X_{b,z}\right).$$

For round $z=0$, $X_{b,0}$ is empty, so $IB_b/\sim X_{b,0} = IB_b$ and therefore $c \in IB_b/\sim X_{b,0}$ for all $c \in IB_b$. Assuming the invariant holds at some round $z < Z$, we now show it holds at round $z+1$. Consider any cell $c \in IB_b$. We must show that

$$\left(\exists c' \in X_{b,z+1} : c' \sim c \wedge c' \leq_{oo(w(b)),t} c\right) \vee \left(c \in IB_b/\sim X_{b,z+1}\right).$$

Case 1: Suppose $\exists c' \in X_{b,z} : c' \sim c \wedge c' \leq_{oo(w(b)),t} c$. Then we are done, because $X_{b,z+1} \supseteq X_{b,z}$.

Case 2: Otherwise, by the invariant for round *z*, we have $c \in IB_b/\sim X_{b,z}$. Let c' be the unique cell in $X_{b,z+1} - X_{b,z}$.

Case 2a: Suppose $c' \sim c$ (which includes the case $c' = c$). Because the global minimum greedy algorithm chooses a most-important not-yet-conflicted cell on each round, we have $c' \leq_{oo(w(b)),t} c$. Thus we have $c' \in X_{b,z+1}$ and $c' \sim c$ and $c' \leq_{oo(w(b)),t} c$, and we are done.

Case 2b: On the other hand, if c' does not conflict with *c*, we have $c \in IB_b/\sim X_{b,z+1}$, and we are done. This completes the proof of the invariant.

By the termination condition, $IB_b/\sim X_{b,Z} = \{\ \}$, the invariant at the end of the final round reduces to

$$\forall c \in IB_b : \exists c' \in X_{b,Z} : c' \sim c \wedge c' \leq_{oo(w(b)),t} c,$$

from which the LOOTFA matching condition follows immediately.

It also turns out that any legal LOOTFA match X'_b can be produced by some run of the global minimum greedy algorithm, by choosing the elements of X'_b in non-decreasing order according to $<_{oo(w(b)),t}$. Hence the possible results of the global minimum greedy algorithm are precisely the matches that are allowed by the definition of a LOOTFA switch.

Note that for each pair of input *i* and output *o* an implementation can ignore any cells in $IB_{b,\text{i}=i,\text{o}=o}$ except for any single minimal element according to $<_t$. If the timestamps for cells of input *i* and output *o* advance monotonically with inhalation time (as in our initial LOOTFA timestamp ordering in Section 3.3), a virtual output queue will always have a most important cell at the front.

6.2. The per-input minimum greedy algorithm

We consider the case of the trivial timestamp ordering, in which all cells tie. This reduces the LOOTFA switch to a LOOFA switch. Observe that if an included cell c' has an output conflict with *c*, that is, $o(c') = o(c)$, then we have $oo_b(c') = oo_b(c)$ and hence $c' =_{oo(w(b)),t} c$. Because of this property, a LOOFA match can be computed by the *per-input minimum greedy* algorithm, in which each round *z* consists of two parts: first choosing an input i arbitrarily, subject to the constraint that $IB_{b,\mathrm{i}=i}/\sim X_{b,z}$ is non-empty; and then producing $X_{b,z+1}$ from $X_{b,z}$ by adding a minimal element of $IB_{b,\mathrm{i}=i}/\sim X_{b,z}$ according to $<_{oo(w(b)),t}$.

Regardless of the order in which the inputs are considered, the per-input minimum greedy algorithm produces a match that satisfies the LOOTFA matching condition. The proof uses the same invariant as used in the previous section,

$$\forall c \in IB_b : \left(\exists c' \in X_{b,z} : c' \sim c \wedge c' \leq_{oo(w(b)),t} c\right) \vee \left(c \in IB_b / \sim X_{b,z}\right).$$

The only difference occurs in the proof within Case 2a that c' is at least as important as c, that is, $c' \leq_{oo(w(b)),t} c$. If c' and c share the same input, then the result follows from the fact that the per-input minimum greedy algorithm chooses a most-important not-yet-conflicted cell from a chosen input. Otherwise, c' and c must share the same output, from which it follows that $c' =_{oo(w(b)),t} c$, as explained above.

The per-input minimum greedy algorithm has a parallel implementation in which each input extends a bid to the destined output of its most important non-conflicted cell, and then each output that gets a bid accepts one and rejects the others. Additional bid-accept rounds handle rejected inputs. Although typically only a few rounds are necessary, in the worst case N rounds are required. Krishna *et al.* [1] use this implementation of the per-input minimum greedy algorithm.

Unfortunately, the per-input minimum greedy algorithm does not work for a non-trivial timestamp ordering, which in general is the case in a LOOTFA switch.

Acknowledgments

The authors would like to thank Greg Nelson for his advice on clarifying our notation and Sharon Perl for many helpful comments on an earlier draft.

References

1. P. Krishna, N. S. Patel, A. Charny, R. Simcoe. “On the Speedup Required for Work-Conserving Crossbar Switches.” Accepted for IWQoS-98.
2. B. Prabhakar and N. McKeown. “On the Speedup Required for Combined Input and Output Queued Switching.” *Computer Systems Lab, Technical Report* CSL-TR-97-738, Stanford University.