Response Time Properties of Some Asynchronous Circuits

Jo Ebergen* and Robert Berks
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
{jebergen,rtberks}@maveric.uwaterloo.ca

Abstract

We discuss response time properties of linear arrays and tree-like arrays of cells with various handshake communication behaviours. The response times of a network are the delays between requests and succeeding acknowledgments for the first cell. We derive simple formulas for the worst-case response time and amortized response time of such networks using a general variable-delay model, where delays may vary between a lower and upper bound. The properties are independent of any particular implementation of the cells of the network.

1. Introduction

We present some properties on the response time of some regular asynchronous networks. The basic problems that we address can be illustrated with the following example.

Consider a linear array of $L+1$ cells as indicated in Figure 1 (where $L = 3$). We are only interested in the communication behaviour at the interface of each cell as an ordering of events. Each interface between two cells has the same communication behaviour, viz., a repetition of a request r followed by an acknowledgment a. A request followed by an acknowledgment is called a handshake. The end cell handshakes at its only interface, the other cells perform handshakes at both interfaces. Each non-end cell in this array has a communication behaviour that synchronizes the handshakes at both sides of the cell in some fashion. One of the simplest synchronizations is that for each handshake on the left there is exactly one handshake on the right.

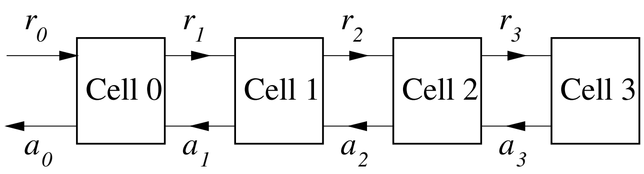

Figure 1. A linear array of cells

In these communication behaviours, the requests and acknowledgments are considered as instantaneous events, and delays can be incurred in the cells or the environment of the pipeline. For each non-end cell we assume that the delays between the receipt of the *last* input and the production of any of the two outputs is always bounded from below by δ and from above by Δ. For the last cell we assume that its delays are bounded from below by δ_L and from above by Δ_L. (There is a reason for distinguishing the last cell from the other cells, which will become clear later.) Finally, we assume that the environment can provide a next request r_0 after an acknowledgment a_0 after a minimum delay of δ. There is no upper bound for the environment delays.

The delays of the cells do not have to be fixed, but may vary between a fixed upper and lower bound. Delays may depend on the data received, or they may vary over different instances of the same cell, or they may vary over time. Furthermore, the delay model does not depend on particular implementations of the basic cells. The only requirement is that the different implementations satisfy the upper bounds and lower bounds for the delays of the basic cells. If we take the upper and lower bound to be equal, as often happens in implementations using data bundling, we get the results for a fixed delay network.

Under these delay assumptions we would like to know what you can say about the worst-case response time of the linear network, that is, what is the maximum delay between any request r_0 and succeeding acknowledgment a_0 over all possible delay distributions? The next question then is whether the worst-case response time depends on the length of the array L or any of the bounds δ, Δ, δ_L, and Δ_L? For example, we show that the worst-case response time for the linear network above is bounded from above

*Currently on leave at Sun Microsystems Laboratories, Mountain View, USA.

by $\Delta_L + (L+1)(\Delta - \delta)$ and that this bound is tight. We also show under which delay distributions this bound can be achieved.

The second problem we consider is calculating an upper bound for the amortized response time of a network. The amortized response time gives an upper bound for the average response time of a network. The amortized response time can differ significantly from the worst-case response time. For example, we show that for the linear network above the amortized response time is bounded from above by $\Delta_L + \Delta - \delta$, which is independent of the length of the array L. This bound is also tight.

We can consider more general behaviours than the one considered above. For example, we can consider cells where for every $n, n > 1$, handshakes with the left neighbour there is at most one handshake with the right neighbour. What can we say about the worst-case response time and amortized response time in these cases? This is the third problem that we address. We show that the following properties hold for the worst-case response time *WR* and amortized response time *AR*.

- $WR \leq \Delta + (L+1)(\Delta - \delta)$ provided that $\Delta_L \leq 2\delta(n^{L-1} - 1) + \Delta$.

  Thus, even if the delay through the last cell increases exponentially with L (as in $2\delta n^L$), the worst-case response time of the network increases at most linearly with L (as in $L(\Delta - \delta)$).

- $AR \leq (3\Delta + \delta)/2$ provided that $\Delta_L \leq 2\delta(n^{L-1} - 1) + \Delta$.

- $WR \leq 2\Delta$ provided that $\Delta_L \leq \Delta n^{L-1} - (L-1)(\Delta - \delta)$ and $\Delta \leq 2\delta$.

In the networks above we have parameterized the maximum and minimum delay for the last cell by Δ_L and δ_L respectively. Thus we can find out to what extent these bounds on the delay of the last cell can influence the response times of the network. Knowing the extent of this influence may give a designer more (or less) freedom in choosing an implementation for the end cell. For example, for linear arrays where $n > 1$ applies, a bounded worst-case and amortized response time is maintained even if the maximum delay Δ_L of the end cell increases exponentially with the length of the array. In such cases, a designer has the freedom to choose a few fast non-end cells and a very slow end cell so that we still achieve a bounded worst-case and bounded amortized response time. For other linear arrays where $n = 1$, the worst-case and amortized response time linearly depend on the maximum delay of the end cell. In these cases it may be important to choose a fast end cell.

The end cell is also a special cell for other reasons. It could be a part of an environment in which the non-end cells are placed. For example, for a simple FIFO the end cell could be seen as the "get" environment. In such cases, the formulas indicate to what extent the delays at the "get" environment can influence the response time at the "put" environment.

Finally, we mention several generalizations of our results, for example to tree-like arrays of cells, and some applications.

Due to a lack of space, some proofs have been omitted. These proofs can be found in [3].

Symbol	Usage
n	Handshake multiplication factor. ($n = 1$ is typical of micropipelines)
δ	Lower bound on non end cell delays and environment delays
Δ	Upper bound on non end cell delays
δ_L	Lower bound on the end cell delay
Δ_L	Upper bound on the end cell delay
L	Number of non end cells and index of end cell
WR	worst-case response time
AR	amortized response time

Table 1. Summary of symbols used.

2. Related Work

The delay assumptions and dependency graphs (also called process graphs) in this paper are inspired by the model used in [9, 10]. In [9, 10] algorithms are described to obtain exact bounds for the delay between two given occurrences of events in a given dependency graph under a variable delay model. A much larger class of networks and communication behaviours is allowed than we use here. In contrast, we have focused on giving formula expressions for a small set of regular process graphs. Such formulas have a few parameters, like the depth of the network and the upper and lower bounds for certain delays. These formulas allow for quick back-of-the-envelope calculations.

More detailed performance analysis and optimization techniques, all the way down to transistor sizing, are given in [2]. Techniques for analyzing the throughput and latency of micropipelines and rings are proposed in [19], where formulas are derived for the throughput as a function of the forward and backward latencies of the stages and the number of tokens in the ring. However, a fixed-delay model is used. See also [15], where these techniques have been applied. A fixed delay model is also used in [13], where an algorithm is presented to compute the average cycle time of arbitrary Signal Graphs. In [8] formulas for upper bounds of the utilization of pipelines are presented under various assumptions for the delay distributions for the latencies of the cells. In [14] yet another model is used for analyzing the response time of implementations. The model is based on the notion of sequence functions and is more restrictive than a variable delay model. Kearney and Bergmann [12] perform an analysis of a pipeline with variable delays and a multiplication factor of $n = 1$. Their results are largely based on simulation, and they do not give closed-form formulas.

3. Communication Behaviours

Consider the cell in Figure 2 from a linear array. At interface i, the communication behaviour for the handshakes can be given by $*[r_i?; a_i!]$. That is, we take an input request, $r_i?$, and then we output an acknowledge $a_i!$. (? denotes input, while ! denotes output.) This behaviour then repeats (indicated by $*[..]$). At interface $i+1$ the communication behaviour is given by $*[r_{i+1}!; a_{i+1}?]$.

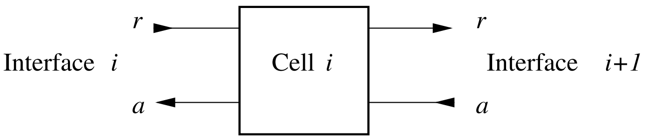

Figure 2. One cell

Cell i synchronizes the handshakes at interface i and $i+1$. In general, the communication behaviour for a cell can take many different forms. We restrict ourselves to behaviours for cells where for every n handshakes on interface i there is one handshake on interface $i+1$. More precisely, the communication behaviours we look at are given by

$$(r_i?;\ *[\ (a_i!;\ r_i?)^{n-1};\ ((a_i!;\ r_i?)||(r_{i+1}!;\ a_{i+1}?))\ ])$$

In other words, only every n-th occurrence of the segment $a_i!;\ r_i?$ takes place in parallel with a segment $r_{i+1}!;\ a_{i+1}?$. Later we consider cases where more parallelism is allowed.

For $n = 1$ we have a communication behaviour that can be represented by

$$(r_i?;\ *[\ (a_i!;\ r_i?)||(r_{i+1}!;\ a_{i+1}?)\ ])$$

An unfolded process graph for $n = 1$ for a cell is given in Figure 3.

Let us denote the k-th occurrence of event a_i by (a_i, k), where $k \geq 0$. A process graph gives a precedence relation for occurrences of events. A precedence between occurrences of events e_0 and e_1 is denoted by $e_0 \rightarrow e_1$. For example, in Figure 3 we have the precedences

$$(r_i, j) \rightarrow (a_i, j) \quad \text{and} \quad (a_{i+1}, j-1) \rightarrow (a_i, j)$$

These precedences indicate that occurrence j of event a_i can only happen after occurrence j of event r_i and occurrence $j-1$ of event a_{i+1} has happened.

Solid arrows indicate "direct" precedence relations brought about by the cell. Dashed arrows indicate "indirect" precedence relations brought about by the environment of the cell.

If we consider the complete network of 3 non-end cells, one end cell, and the environment of the pipeline, then we can depict its behaviour in a process graph (Figure 4), where all edges represent direct precedences brought about by some component in the network. The dashed arrows represent direct precedences caused by the environment of the pipeline (top) or by the end cell (bottom).

For $n = 2$ we have a communication behaviour that can be represented by

$$(r_i?;\ *[\ a_i!;\ r_i?;\ ((a_i!;\ r_i?)||(r_{i+1}!;\ a_{i+1}?))\ ])$$

An unfolded process graph for $n = 2$ for a cell is given in Figure 5 and the corresponding graph for a linear array of four cells is given in Figure 6. For later use we have labelled the nodes in this graph with occurrence indexes.

4. The Delay Model

Once we have a process graph we can assign a time stamp to each node in the graph, where nodes are occurrences of events. This timing assignment, T, to nodes in the graph has to satisfy certain restrictions given by the delay model. The delay restrictions translate into lower and upper bounds for delays that can be incurred in each arc of the process graph and into the times that can be assigned to a node.

Let c be a node in a (unfolded) process graph with c an output of a non-end cell. Our delay model prescribes that any timing assignment T of the process graph must satisfy the following constraints.

$$T_{pred}(c) + \delta \leq T(c) \leq T_{pred}(c) + \Delta$$

where $T(c)$ is the time assigned to node c and

$$T_{pred}(c) = \max\{T(b) \mid b \text{ is a direct predecessor of c}\}$$

This condition stipulates that the delay is always measured from the time $T_{pred}(c)$, when the *last* directly preceding event occurs, to the actual occurrence of the event c. The constraints prescribe that all delays through non-end cells have upper bound Δ and lower bound δ, that is

$$\delta \leq \text{Non end cell delays} \leq \Delta$$

The delays incurred in the end cell can be expressed as $T(a_L, j) - T(r_L, j)$ for occurrence j, $j \geq 0$, of output a_L

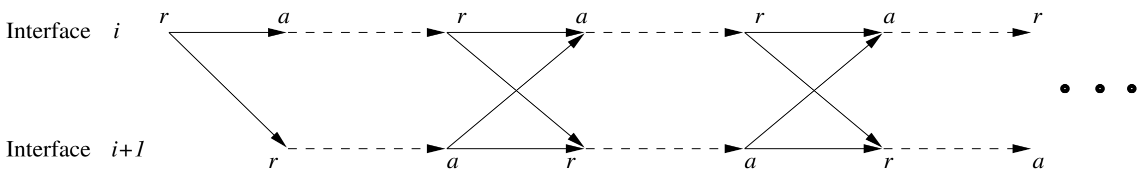

Figure 3. Unfolded communication behaviour for a cell

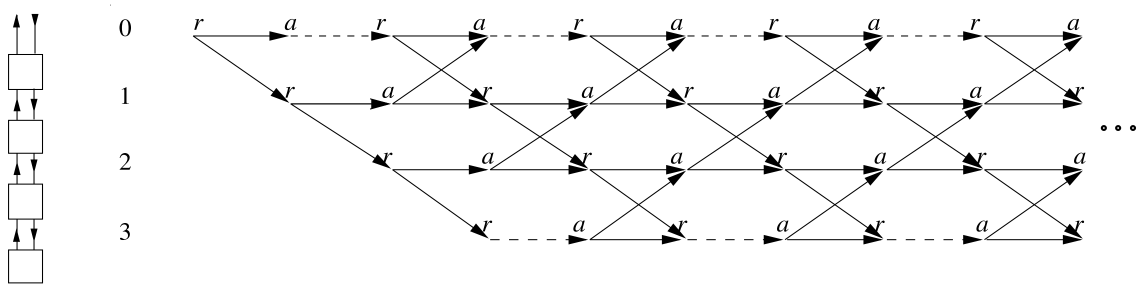

Figure 4. Unfolded communication behaviour for a linear array of four cells and $n = 1$

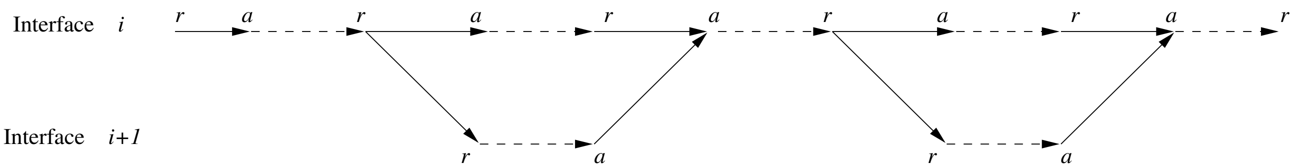

Figure 5. Unfolded communication behaviour for a cell with $n = 2$

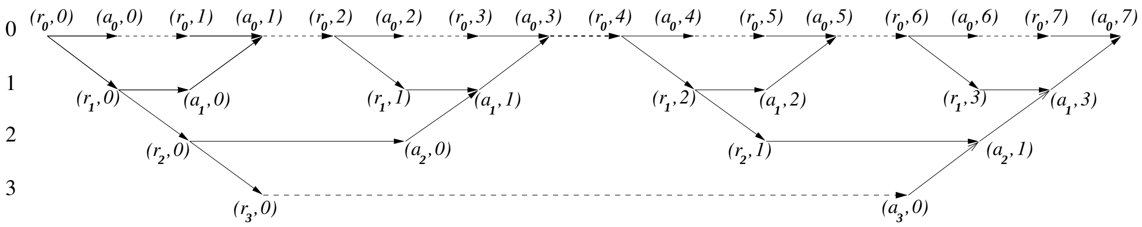

Figure 6. Unfolded process graph for a linear array of four cells with $n = 2$

and input r_L. We assume that the delays of the end cell have a lower bound δ_L and upper bound Δ_L, that is

$$\delta_L \leq \text{End cell delays} \leq \Delta_L$$

Furthermore we assume that $\delta \leq \delta_L$ and $\Delta \leq \Delta_L$.

Finally, we assume that the delay through the environment of the pipeline is at least δ, that is, $T(r_0, j+1) - T(a_0, j) \geq \delta$ for all $j \geq 0$.

When a dependency graph and a timing assignment is given, we can define the useful notion of *critical paths*. We define an edge $a \to b$ to be a *critical edge* if $\delta \leq T(b) - T(a) \leq \Delta$ for b an output of a non-end cell, or if $\delta_L \leq T(b) - T(a) \leq \Delta_L$ for b an output of an end cell. Note that the lower bound δ is automatic by our statement that edge delays must be at least δ. The upper bound may not always be true, because other dependencies may make event b occur later. We call a directed path a *critical path*, if all its edges are critical edges.

We can view the critical edges as pieces of elastic which can "stretch" between lengths δ and Δ, or δ_L and Δ_L. Note that for each node at least one incoming edge must be a critical edge, because of the delay constraints. If we stretch an edge beyond Δ it breaks. On a critical path, we know that no bit of elastic on that path is broken, since, by the choice of the edges on the path, none of the bits of elastic is stretched beyond its upper bound Δ or Δ_L.

5. A Timing Property

Before we formulate some properties about response times, we define the concept of *response depth* of an occurrence of an event. Let the following path exist in a process graph for a behaviour B

$$(a_i, j_i) \to (a_{i-1}, j_{i-1}) \to \ldots(a_{k+1}, j_{k+1}) \to (a_k, j_k)$$

with $i \geq k$ and occurrences $j_i < j_{i-1} < j_{i-2} < \ldots < j_{k+1} < j_k$. That is, the acknowledgment a_k depends on a_i via a sequence of acknowledgments, one for each interface. Let furthermore T be a timing assignment for behaviour B. We say that (a_k, j_k) has *response depth* $i-k$ for behaviour B and timing assignment T iff the path $(a_i, j_i) \to (a_{i-1}, j_{i-1}) \to \ldots(a_{k+1}, j_{k+1}) \to (a_k, j_k)$ is a critical path, and the edge $(a_{i+1}, j_{i+1}) \to (a_i, j_i)$ does not exist, or, if it does exist, it is not critical. We call this critical path the "response path" for acknowledgment (a_k, j_k). For example, if we have a response path with $k = 0$, and $i = L$, then we have a critical path of acknowledgments from the end-cell to the environment, with response depth L.

We now state the main theorem.

Theorem 1 *If the response depth of (a_i, j_i) is $k, 1 \leq k \leq L-i$, then*

$$T(a_i, j_i) - T(a_i, j_i - n^{k-1}) \leq C_{i+k} + \Delta + k(\Delta - \delta)$$

where

$$C_k = \begin{cases} \Delta & \text{if } 0 \leq k < L \\ \Delta_L & \text{if } k = L \end{cases}$$

□

This theorem says that if acknowledgment (a_i, j_i) has a response depth of $k, 1 \leq k \leq L-i$, then the duration of the n^{k-1} handshake cycles at interface i that end in (a_i, j_i) is at most $C_{i+k} + \Delta + k(\Delta - \delta)$. The factor n^{k-1} in the above equation will prove useful, because we give a linear bound to the duration between two events that have an exponential number of events between them (if $n > 1$).

Here is a brief example of the main theorem. Suppose $n = 2$, $k = 2$, $L = 3$, and that we are looking at event a_0 ($i = 0$). Refer to Fig 6. We will choose $j_0 = 7$. Therefore we wish to show that

$$\begin{aligned} T(a_0,7) - T(a_0, 7 - 2^{(2-1)}) &\leq C_2 + \Delta + 2(\Delta - \delta) \\ T(a_0,7) - T(a_0, 7-2) &\leq \Delta + \Delta + 2(\Delta - \delta) \\ T(a_0,7) - T(a_0,5) &\leq 4\Delta - 2\delta \end{aligned}$$

Since $k = 2$, we know that $(a_0, 7)$ is dependent on an acknowledgment 2 levels deeper in the pipeline, viz., $(a_2, 1)$, and not on acknowledgments any deeper. Therefore we can ignore the dependency from $(a_3, 0)$. So, tracing further back, $(a_0, 7)$ must be dependent on $(r_1, 2)$. The path from $(r_1, 2)$ to $(a_0, 7)$ cannot be longer than 4Δ. The path from $(r_1, 2)$ to $(a_0, 5)$ cannot be shorter than 2δ. So we get

$$\begin{aligned} & T(a_0,7) - T(a_0, 7-2^1) \\ \leq \quad & (T(r_1,2) + 4\Delta) - (T(r_1,2) + 2\delta) \end{aligned}$$

which is equivalent to

$$T(a_0,7) - T(a_0,5) \leq 4\Delta - 2\delta$$

Before we prove the main theorem, we give some lemmas.

Lemma 2 *If we have the direct dependency $b \to c$, then*

$$T(c) - T(b) \geq \delta \quad \textit{and, hence } T(b) - T(c) \leq -\delta$$

Lemma 3 *For any $k, 0 \leq k \leq L$, if the response depth of (a_k, j_k) is 0, then*

$$T(a_k, j_k) - T(r_k, j_k) \leq C_k$$

Lemma 4 *If the dependency $(a_{i+1}, j_{i+1}) \to (a_i, j_i)$ exists, and if $(a_{i+1}, j_{i+1} - n^{k-1})$ exists for some $k > 0$ (where for our purposes the event exists if the occurrence index $j_{i+1} - n^{k-1} \geq 0$). Then the dependency $(a_{i+1}, j_{i+1} - n^{k-1}) \to (a_i, j_i - n^k)$ also exists.*
□ *(Proofs omitted).*

As an example of Lemma 4, we look at Fig 6. We set $n = 2$, $k = 2$, and $i = 0$, and choose $j_i = 7$ and $j_{i+1} = 3$.

$$\begin{array}{rcl} (a_{i+1}, j_{i+1}) & \rightarrow & (a_i, j_i) \\ = (a_1, 3) & \rightarrow & (a_0, 7) \end{array}$$

Given this dependency and that $(a_{i+1}, j_{i+1} - n^{k-1})$ $\{= (a_1, 1)\}$) exists we have

$$\begin{array}{rcl} (a_{i+1}, j_{i+1} - n^{k-1}) & \rightarrow & (a_i, j_i - n^k) \\ = (a_1, 3 - 2^1) & \rightarrow & (a_0, 7 - 2^2) \\ = (a_1, 1) & \rightarrow & (a_0, 3) \end{array}$$

which is indeed correct.

Lemma 2 says that if we know of the existence of a dependency, then the separation between the source and target event must be at least δ. Lemma 3 says that if an acknowledgment to a request to cell k, depends only on cell k itself and not on cells $k+1, \ldots L$, then the only delay incurred is the delay of cell k. Lemma 4 says that the occurrence index changes by a factor of n between levels when we keep the dependency pattern the same. In the proof below, we assume that the occurrence indices are large enough to satisfy the second condition of Lemma 4.

With the preceding lemmas we can now prove the main theorem.

Proof of Theorem 1. By induction on k.
Basis. Assume that (a_i, j_i) has a response depth of $k = 1$. Let $(a_{i+1}, j_{i+1}) \rightarrow (a_i, j_i)$, then (a_{i+1}, j_{i+1}) has response depth 0. We have a dependency graph as in Figure 7. From

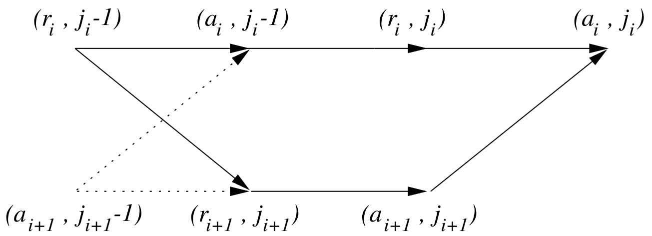

Figure 7. A part of the behaviour graph. Dotted edges may not be present if $n > 1$.

the definition of $T_{pred}(c)$ and the dependency graph in Figure 7, we have $T_{pred}(r_{i+1}, j_{i+1}) = T_{pred}(a_i, j_i - 1)$. In the proof below we take $T_{pred} = T_{pred}(a_i, j_i - 1)$. We wish to derive an upper bound for $T(a_i, j_i) - T(a_i, j_i - 1)$. We observe

$T(a_i, j_i) - T(a_i, j_i - 1)$

$=$ { Break path into 4 edges. }

$(T(a_i, j_i) - T(a_{i+1}, j_{i+1})) +$
$(T(a_{i+1}, j_{i+1}) - T(r_{i+1}, j_{i+1})) +$
$(T(r_{i+1}, j_{i+1}) - T_{pred}) +$
$(T_{pred} - T(a_i, j_i - 1))$

$\leq$ { (a_i, j_i) has positive response depth so timing dependency exists. Also, ack/ack pair cannot be environment or end-cell delays }

$\Delta + (T(a_{i+1}, j_{i+1}) - T(r_{i+1}, j_{i+1})) +$
$(T(r_{i+1}, j_{i+1}) - T_{pred}) +$
$(T_{pred} - T(a_i, j_i - 1))$

$\leq$ { (a_{i+1}, j_{i+1}) has resp. depth 0, Lemma 3 }

$\Delta + C_{i+1} + (T(r_{i+1}, j_{i+1}) - T_{pred}) +$
$(T_{pred} - T(a_i, j_i - 1))$

$=$ { $T_{pred} = T_{pred}(a_i, j_i - 1) = T_{pred}(r_{i+1}, j_{i+1})$ }

$\Delta + C_{i+1} + (T(r_{i+1}, j_{i+1}) - T_{pred}(r_{i+1}, j_{i+1})) +$
$(T_{pred} - T(a_i, j_i - 1))$

$\leq$ { $\delta \leq$ cell delays $\leq \Delta$, Lemma 2 }

$\Delta + C_{i+1} + \Delta + (-\delta)$

$=$ { calc. }

$C_{i+1} + \Delta + (\Delta - \delta)$

Step. Assume the theorem holds for $k > 0$ and that (a_i, j_i) has response depth $k+1$. Let $(a_{i+1}, j_{i+1}) \rightarrow (a_i, j_i)$, then (a_{i+1}, j_{i+1}) has response depth k. We observe

$T(a_i, j_i) - T(a_i, j_i - n^k)$

$=$ { Break path into three: edge, path, edge. }

$((T(a_i, j_i)) - T(a_{i+1}, j_{i+1})) +$
$(T(a_{i+1}, j_{i+1}) - T(a_{i+1}, j_{i+1} - n^{k-1})) +$
$(T(a_{i+1}, j_{i+1} - n^{k-1}) - T(a_i, j_i - n^k))$

$\leq$ { (a_{i+1}, j_{i+1}) has resp. depth k, induction hyp. }

$((T(a_i, j_i)) - T(a_{i+1}, j_{i+1})) +$
$C_{i+k+1} + \Delta + k(\Delta - \delta) +$
$(T(a_{i+1}, j_{i+1} - n^{k-1}) - T(a_i, j_i - n^k))$

$\leq$ { (a_i, j_i) has positive response depth }

$\Delta + C_{i+k+1} + \Delta + k(\Delta - \delta) +$
$(T(a_{i+1}, j_{i+1} - n^{k-1}) - T(a_i, j_i - n^k))$

$\leq$ { By Lemma 4, $(a_{i+1}, j_{i+1} - n^{k-1}) \rightarrow (a_i, j_i - n^k)$ exists and we can apply Lemma 2 }

$\Delta + C_{i+k+1} + \Delta + k(\Delta - \delta) + (-\delta)$

$\leq$ { calc. }

$C_{i+k+1} + \Delta + (k+1)(\Delta - \delta)$

$\square$

6. Worst-Case Response Time

From the last theorem we can infer a number of interesting properties. We can look at worst-case cycle time and worst-case response time. Worst-case cycle time is the largest value that $T(a_0, j+1) - T(a_0, j)$ can take over all possible values of j and all valid delay distributions. The worst-case cycle time is not that interesting in this delay model, since the environment of the pipeline can stretch the cycle time as much as possible by waiting to send a request. (The environment delays do not have an upper bound.) The worst-case response time WR is more interesting. We have the following theorem.

Theorem 5 *The worst-case response time, WR, for any pipeline (as defined in Section 3 operating under a delay model of Section 4) is bounded from above as*

$$\begin{aligned} WR \quad \leq \quad & \max(\{C_k + \Delta + k(\Delta - \delta) - 2\delta n^{k-1} + \delta \\ & \mid 1 \leq k \leq L\} \cup \{C_0\}) \end{aligned}$$

Proof. We have to find an upper bound for $T(a_0, j) - T(r_0, j)$ over all $j \geq 0$ and all valid delay distributions. Each acknowledgment has a response depth k, where $0 \leq k \leq L$. For a response depth $k = 0$, we have from Lemma 2, $T(a_0, j) - T(r_0, j) \leq C_0$. For a response depth of $k > 0$, we use the result of the previous Section. We observe

$$\begin{aligned} & T(a_0, j) - T(r_0, j) \\ = \quad & \{ \text{ calc. } \} \\ & (T(a_0, j) - T(a_0, j - n^{k-1})) + \\ & (T(a_0, j - n^{k-1}) - (T(r_0, j)) \\ \leq \quad & \{ \ (a_0, j) \text{ has response depth } k \text{, Theorem 1 } \} \\ & C_k + \Delta + k(\Delta - \delta) + (T(a_0, j - n^{k-1}) - (T(r_0, j)) \end{aligned}$$

Notice that there are n^{k-1} handshakes at level 0 between occurrences (a_0, j) and $(a_0, j - n^{k-1})$. The handshakes at level 0 experience delays through the environment and through cell 0. Assuming that the environment's and the cell's minimum response time is δ, the minimum duration of one cycle through environment and cell is 2δ. Therefore the minimum duration from $T(a_0, j - n^{k-1})$ to $T(r_0, j)$ is $2\delta n^{k-1} - \delta$. This then leads to the inequality

$$T(a_0, j) - T(r_0, j) \leq C_k + \Delta + k(\Delta - \delta) - 2\delta n^{k-1} + \delta$$

Maximizing these upper bounds over all $k, 0 \leq k \leq L$ gives the desired result.
□

The above theorem holds for all pipelines with a multiplication factor $n > 0$. The case when $L = 0$ is not especially interesting, as this is simply an environment and an end cell. In this case the formula resolves to $WR \leq \Delta_L$. For $L > 0$, we consider two special cases: $n = 1$ and $n > 1$. Instantiating the above theorem for $n = 1$, we derive

Theorem 6 *The worst-case response time WR for a linear array with multiplication factor $n = 1$ satisfies*

$$WR \leq \Delta_L + (L+1)(\Delta - \delta)$$

□ *(Proof omitted).*

In other words, for a pipeline with multiplication factor 1, the worst-case response time depends linearly on Δ_L, L, and $\Delta - \delta$. With the proper distribution of delays, this upper bound can indeed be attained. See Figure 8. All solid thin lines have a delay of δ, and all solid thick lines have a delay of Δ. The dashed, thick line has a delay of Δ_L. The dashed, thin line has a delay δ_L. The delay between $(r_0, 4)$ and $(a_0, 4)$ is $\Delta_L + 4(\Delta - \delta)$. Obviously, this example can be generalized for a pipeline of any length.

For the case where the multiplication factor $n > 1$, we take a function D_L to serve as an upper bound for the delay through the end cell. Let

$$D_L = 2\delta(n^{L-1} - 1) + \Delta$$

for $1 \leq L$. If we assume that $\Delta_L \leq D_L$, we derive with Theorem 5

Theorem 7 *The worst-case response time WR for a linear array with multiplication factor $n \geq 1$ satisfies*

$$WR \leq \Delta + (L+1)(\Delta - \delta)$$

provided the delay through the end cell satisfies

$$\Delta_L \leq 2\delta(n^{L-1} - 1) + \Delta$$

□ *(Proof omitted).*

In words, if the delay in the last cell increases at most exponentially with L (as in $2\delta n^L$), the worst-case response time of the network increases at most linearly with L (as in $k(\Delta - \delta)$).

A special case occurs when the delays in the cells are fixed ($\Delta = \delta$), which often happens in bundled data implementations where matching delays are fixed, then the upper bound for the worst-case response time is bounded by Δ.

If the upper and lower bounds for the delays differ, we can still maintain a bounded response time for $n > 1$ if we are willing to let the delay in the last cell increase a little bit less than D_L. How much less increase is needed depends on the ratio of Δ and δ. For example, we have

Theorem 8 *If $\Delta \leq 2\delta$ then the worst-case response time WR for a linear array with multiplication factor $n > 1$ satisfies*

$$WR \leq 2\Delta$$

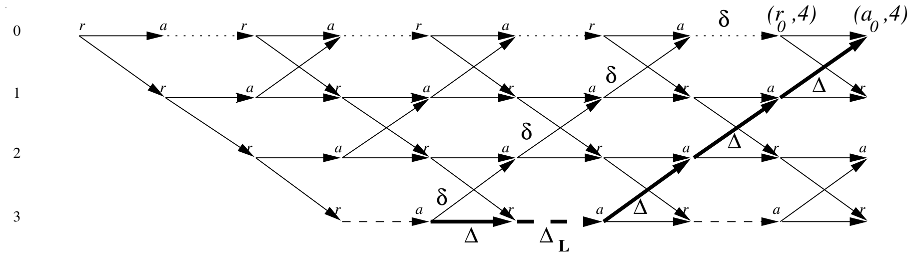

Figure 8. Example with worst-case response

provided the delay through the end cell satisfies

$$\Delta_L \leq \Delta n^{L-1} - (L-1)(\Delta - \delta)$$

□ *(Proof omitted).*

7. Amortized Response Time

The amortized response time is the average response time under the worst-case delay distribution. In other words, the average response time for any delay distribution is at most the amortized response time. We show that the amortized response time is bounded by constants independent of L for pipelines with multiplication factor $n > 1$, as we would expect, but also for pipelines with $n = 1$. In other words, although the worst-case response time for a pipeline with $n = 1$ depends linearly on L, "on average" the response time for such a pipeline does not depend on L.

We consider two cases $n = 1$ and $n > 1$ and use the notion of a critical response path to prove an upper bound for the amortized response time for these pipelines. Here is the theorem for the case $n = 1$.

Theorem 9 *The amortized response time AR of a pipeline with multiplication factor $n = 1$ is bounded from above as*

$$AR \leq \Delta_L + \Delta - \delta$$

Proof. Consider a pipeline with $n = 1$ and a timing assignment for its dependency graph. Take an occurrence of an acknowledgment, say (a_0, j_0). From acknowledgment (a_0, j_0) there is a critical path to some request (r_0, i_0) with $i_0 \leq j_0$ and on this critical path there is no other event at level 0. Let the length of this path be h. Notice that if $h = 1$, then $j_0 = i_0$ and we have $T(a_0, j_0) - T(r_0, i_0) \leq \Delta$. For a path of length h, a delay of Δ_L can be obtained at most in every other edge of the path and only in edges at level L. In the other edges a delay of at most Δ can be obtained. Let $h = 2k + 1$ for some $k \geq 0$. For $k < L$ we then get an upper bound of

$$T(a_0, j_0) - T(r_0, i_0) \leq (2k + 1)\Delta$$

Furthermore, between (r_0, i_0) and (a_0, j_0) there are $k = (h - 1)/2$ handshake cycles plus one cell delay. (A handshake cycle consists of one environment and one cell delay.) Environment delays are at least δ. In k handshake cycles, the total time for the environment delays is at least $k\delta$. Hence, the total time for the pipeline responses between $T(r_0, i_0)$ and $T(a_0, j_0)$ is at most $(2k + 1)\Delta - k\delta$. So the average response time AR_k for the pipeline responses between $T(r_0, i_0)$ and $T(a_0, j_0)$ satisfies

$$AR_k \leq ((2k{+}1)\Delta{-}k\delta)/(k{+}1) = 2\Delta{-}\delta{-}(\Delta{-}\delta)/(k{+}1)$$

for $0 \leq k < L$. Maximizing this upper bound over all $k, 0 \leq k < L$, yields an upperbound for the amortized response time, $AR0$.

$$AR0 \leq 2\Delta - \delta - (\Delta - \delta)/L$$

For $k \geq L$ we get a critical response path of length $2k + 1$ with maximum delay if there are $2L$ edges with delay Δ, $k - L$ edges with delay Δ, and $k + 1 - L$ edges with delay D.

$$T(a_0, j_0) - T(r_0, i_0) \leq 2L\Delta + \Delta_L + (k - L)(\Delta_L + \Delta)$$

Using a similar reasoning as above, we find the amortized response time, AR_k.

$$\begin{aligned}
& AR_k \\
\leq \quad & \{ \text{ From above } \} \\
& (2L\Delta + \Delta_L + (k - L)(\Delta_L + \Delta) - k\delta)/(k + 1) \\
= \quad & \{ \text{ calc. } \} \\
& \Delta_L + \Delta - \delta - (\Delta - \delta + L(\Delta_L - \Delta))/(k + 1)
\end{aligned}$$

Maximizing this upper bound over all $k \geq L$, and taking into account that $\Delta \geq \delta$ and $\Delta_L \geq \Delta$, yields an upperbound for the amortized response time, $AR1$.

$$AR1 \leq \Delta_L + \Delta - \delta$$

The maximum of $AR0$ and $AR1$ gives as upper bound for the amortized response time AR.

$$AR \leq \Delta_L + \Delta - \delta$$

□

Without proof we mention that this bound is tight.

For a multiplication factor $n > 1$ we can use the following theorem.

Theorem 10 *The amortized response time AR of a pipeline with multiplication factor $n > 1$ is bounded from above as*

$$AR \leq (3\Delta + \delta)/2$$

provided the delay in the last cell satisfies

$$\Delta_L \leq 2\delta(n^{L-1} - 1) + \Delta$$

□ *(Proof omitted)*

In other words, if the delay through the last cell increases at most exponentially with L (as in $2\delta n^L$), then the amortized response time for a pipeline with multiplication factor $n > 1$ is bounded by $(3\Delta + \delta)/2$. This bound is not tight.

8. Generalizations and Applications

There are many pipeline designs for various computations all with multiplication factor $n = 1$ and where data is flowing in one or both directions, see [4, 7, 11, 14, 16, 18] for example. If we know what the values are for each of the parameters for certain implementations, we can quickly determine what the upper bounds for the worst-case and amortized response times are. If we know that delays are fixed, that is, $\Delta = \delta$, then we get a worst-case and amortized response time that are bounded.

Without making significant changes to the proofs of the lemmas and the main theorem, our results still apply to other regular networks. Here are some examples.

Consider a linear array where each of the non-end cells satisfies a communication behaviour that allows for more parallelism.

$$(r_i?;\ *[\ (a_i!;\ r_i?)^n \| (sr_{i+1}!;\ sa_{i+1}?)\ ])$$

Here a *all* n handshakes on the left (instead of only one handshake) can take place in parallel with one handshake on the right. In this case the same results apply with respect to WR and AR, where for $n > 1$ the bounds can even be made tighter.

As another example of a generalization, consider a communication behaviour where for every $n > 1$ handshakes on the left there is *at most* one handshake on the right (instead of *exactly* one). The communication behaviours can be given by

$$\begin{array}{l} (r_i?; *[(a_i!; r_i?)^{n-1}; \\ \qquad \mathbf{if}\ B_i\ \mathbf{then}\ (a_i!;\ r_i?) \| (r_{i+1}!;\ a_{i+1}?) \\ \qquad\qquad \mathbf{else}\ (a_i!;\ r_i?) \\ \qquad \mathbf{fi}\ ]) \end{array}$$

Here the guard B_i may depend on data values that have been communicated. If $B_i = \mathbf{true}$, a handshake takes place with the right-hand neighbour; if $B_i = \mathbf{false}$, no handshake takes place with the right-hand neighbour. Also in this generalization the same results apply with respect to WR and AR. Designs that employ a linear array and a multiplication factor of 2 are, for example, modulo-N counter implementations [5, 17] and up-down N counter implementations [6].

If the initial states of the communication behaviour are different, the AR results still apply as given. The WR results also apply, but only after a short period of transitory behaviour.

Finally we mention that the results for *WR* also generalize to tree-like networks. For example, consider a tree where each internal cell for each handshake with its parent has one handshake with one of its children. Furthermore, the handshakes with the children alternate between left and right child (for a binary tree). The communication behaviours are given by

$$\begin{array}{l} (r_i?;\ *[\ (a_i!;\ r_i?) \| (rl_{i+1}!;\ al_{i+1}?); \\ \qquad (a_i!;\ r_i?) \| (rr_{i+1}!;\ ar_{i+1}?)\ ]) \end{array}$$

The terminals rl and al connect the internal node with its left child, the terminals rr and ar with its right child. In this case the results with multiplication factor $n = 2$ apply for worst-case response time, since the communication behaviour that remains if only the handshakes with respect to r, a and (any) one of the children is considered, is the communication behaviour for a linear array with multiplication factor $n = 2$. For example, when the handshakes with the left child are deleted we get

$$(r_i?;\ *[\ (a_i!;\ r_i?);\ (a_i!;\ r_i?) \| (rr_{i+1}!;\ ar_{i+1}?)\ ])$$

In [1] an implementation is given of a FIFO that uses cells with the above communication behaviour.

The amortized behaviour of trees is more complex and is not dealt with in this paper.

9. Concluding Remarks

Given a linear pipeline, various communication behaviours, and bounds on the delays of the cells, we have given formulas for the amortized and worst-case response times of the pipeline. Notably, we have shown that we can achieve bounded amortized response times independent of the length of the pipeline. Furthermore, we have shown under what conditions a bounded worst-case response time can be achieved.

Not only do these formulas allow for quick back-of-the-envelope calculations, but they also allow for a better understanding of which parameters are influential and in what sense they influence the response time of the design. For example, we have shown how slow the end cell can be without affecting the response time. Such knowledge can be exploited conveniently in the design of the last cell by trading speed for better energy efficiency, for example.

The results can be generalized to different behaviours and architectures such as trees. Further generalization would be possible if we would distinguish more parameters, such as forward and backward latencies, instead of just one latency (that is, one cell delay).

10. Acknowledgements

The authors are most grateful for the detailed and useful comments of the reviewers. The second author would like to thank Ed Carr for his helpful comments involving the proofs.

References

[1] E. Brunvand. Low latency self-timed flow through fifos. In *16th Conference on Advanced Research in VLSI*, Mar. 1995.

[2] S. M. Burns and A. J. Martin. Performance analysis and optimization of asynchronous circuits. In C. H. Séquin, editor, *Advanced Research in VLSI: Proceedings of the 1991 UC Santa Cruz Conference*, pages 71–86. MIT Press, 1991.

[3] J. Ebergen and R. Berks. Response time properties of some asynchronous circuits. Technical Report TR-97-02, University of Waterloo, Department of Computer Science, Jan. 1997.

[4] J. Ebergen and R. Hoogerwoord. A derivation of a serial-parallel multiplier. *Science of Computer Programming*, 15:201–215, 1990.

[5] J. C. Ebergen and A. M. G. Peeters. Design and analysis of delay-insensitive modulo-N counters. *Formal Methods in System Design*, 3(3):211–232, Dec. 1993.

[6] J. C. Ebergen, J. Segers, and I. Benko. Parallel program and asynchronous circuit design. In G. Birtwistle and A. Davis, editors, *Proceedings Banff VIII Workshop: Asynchronous Digital Circuit Design*, Workshops in Computing. Springer-Verlag, 1995.

[7] S. Furber. Computing without clocks: Micropipelining the ARM processor. In G. Birtwistle and A. Davis, editors, *Proceedings Banff VIII Workshop: Asynchronous Digital Circuit Design*, Workshops in Computing. Springer-Verlag, 1995.

[8] M. R. Greenstreet and K. Steiglitz. Bubbles can make self-timed pipelines fast. *Journal of VLSI Signal Processing*, 2(3):139–148, Nov. 1990.

[9] H. Hulgaard, S. Burns, T. Amon, and G. Borriello. An algorithm for exact bounds on the time separation of events in concurrent systems. Technical Report TR-94-02-02, University of Washington, Department of Computer Science and Engineering, Feb. 1994.

[10] H. Hulgaard and S. M. Burns. Bounded delay timing analysis of a class of CSP programs with choice. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 2–11, Nov. 1994.

[11] M. B. Josephs, P. G. Lucassen, J. T. Udding, and T. Verhoeff. Formal design of an asynchronous DSP counterflow pipeline: A case study in handshake algebra. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 206–215, Nov. 1994.

[12] D. Kearney and N. W. Bergman. Performance evaluation of asynchronous logic pipelines with data dependant processing delays. In *Second Working Conference on Asynchronous Design Methodologies*, pages 4–13. IEEE Computer Society Press, May 1995.

[13] C. D. Nielsen and M. Kishinevski. Performance analysis based on timing simulation. In *Proc. ACM/IEEE Design Automation Conference*, pages 70–76, June 1994.

[14] M. Rem. Trace theory and systolic computations. In J. W. de Bakker, A. J. Nijman, and P. C. Treleaven, editors, *PARLE: Parallel Architectures and Languages Europe, Vol. I*, volume 258 of *Lecture Notes in Computer Science*, pages 14–33. Springer-Verlag, 1987.

[15] J. Sparsø and J. Staunstrup. Design and performance analysis of delay insensitive multi-ring structures. In *Proc. Hawaii International Conf. System Sciences*, volume I, pages 349–358. IEEE Computer Society Press, Jan. 1993.

[16] I. E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, June 1989.

[17] K. van Berkel. VLSI programming of a modulo-N counter with constant response time and constant power. In S. Furber and M. Edwards, editors, *Asynchronous Design Methodologies*, volume A-28 of *IFIP Transactions*, pages 1–11. Elsevier Science Publishers, 1993.

[18] K. van Berkel and M. Rem. VLSI programming of asynchronous circuits for low power. In G. Birtwistle and A. Davis, editors, *Proceedings Banff VIII Workshop: Asynchronous Digital Circuit Design*, Workshops in Computing. Springer-Verlag, 1995.

[19] T. E. Williams. Analyzing and improving the latency and throughput performance of self-timed pipelines and rings. In *Proc. International Symposium on Circuits and Systems*, May 1992.